連合学習におけるデータ再構築攻撃に対処する要素毎のしきい値Secure Aggregation

水門 巧実[1,a)]　小泉 佑揮[1,b)]　武政 淳二[1,c)]　長谷川 亨[2,d)]

概要：連合学習は生データを共有せずにモデル更新のみをサーバに交換することで協調学習を実現するが，更新から元データが再構築される可能性が指摘されている．Secure Aggregation (SecAgg) は更新の集約のみをサーバに公開することでこの問題に対処するが，ある要素に単一の参加者しか非ゼロ値を提供しないようなスパースな更新では，集約結果から単一参加者の非ゼロ要素が露呈し，データ再構築攻撃に悪用される脆弱性がある．本稿では，少なくとも t 人の参加者が貢献した要素のみ集約値を開示するPer-elementSecAgg を提案する．本手法は標準的な暗号技術のみに依存するため既存 SecAgg との互換性を維持できる．実験評価により，計算・通信コストの増加は限定的であり，モデル性能への影響も小さいことも示され，その実用性が確認された．

キーワード：連合学習，Secure Aggregation，データ再構築攻撃

Per-element Threshold Secure Aggregation against Data Reconstruction Attacks in Federated Learning

Takumi Suimon[1,a)]　Yuki Koizumi[1,b)]　Junji Takemasa[1,c)]　Toru Hasegawa[2,d)]

Abstract: Federated learning enables collaborative training without sharing raw data by exchanging only model updates, but prior studies have shown that raw data can still be reconstructed from individual updates. Secure Aggregation (SecAgg) addresses this issue by ensuring that the server learns only aggregated updates, yet it remains vulnerable under sparse updates: a non-zero element contributed by a single participant can be exposed and exploited for data reconstruction attacks. In this paper, we propose Per-element SecAgg, which reveals an aggregate value only when at least t participants contribute non-zero elements. Our approach relies solely on standard cryptographic primitives, thereby maintaining compatibility with existing SecAgg protocols. Through experiments, we demonstrate that the additional computation and communication overhead is limited, the impact on model performance is small, and the protocol is practical.

Keywords: Federated learning, Secure Aggregation, Data reconstruction attacks

1. はじめに

連合学習 [1] は，データを分散保持したまま参加者が協調的にモデルを学習する手法である．各ラウンドで参加者はグローバルモデルを自身のデータで学習し，更新をサーバに送信する．サーバは更新を集約して新たなモデルを導出し，再び参加者に配布する．連合学習は生データを直接共有しない点をプライバシの根拠とするが，サーバが更新

[1] 大阪大学 大学院情報科学研究科
Graduate School of Information Science and Technology, The University of Osaka, Japan
[2] 島根大学 材料エネルギー学部
Faculty of Materials for Energy, Shimane University, Japan
[a)] t-suimon@ist.osaka-u.ac.jp
[b)] ykoizumi@ist.osaka-u.ac.jp
[c)] j-takemasa@ist.osaka-u.ac.jp
[d)] t_hasegawa@mat.shimane-u.ac.jp

表 1　本稿で頻繁に使用される表記

表記	説明
$\mathcal{C}, \mathcal{D}$	参加者の集合と復号者の集合
δ_D	復号者の離脱率
η_C, η_D	参加者の共謀率と復号者の共謀率
$\boldsymbol{x}_i$	参加者 $i \in \mathcal{C}$ のモデル更新ベクトル
$\mathcal{K}$	モデル更新ベクトル $\boldsymbol{x}_i$ のインデックス集合
t, ℓ	SecAgg のしきい値と秘密分散のしきい値

からデータを再構成できることが指摘されている [2].

Secure aggregation (SecAgg) [3], [4], [5] は，この脅威への主要対策である．SecAgg は少なくとも t 個 ($t \geq 2$) の更新の集約のみをサーバに公開する手法である．これによりサーバは個別更新に直接アクセスできない．

しかし，SecAgg は個別ベクトルは秘匿するが，要素単位の秘匿は保証していない．すなわち，ある要素でただ一人の参加者のみ非ゼロ値を貢献する場合，その値は集約から直接取得できる．近年の研究 [6], [7], [8], [9], [10] は，悪意あるサーバが配布モデルを改竄し，他参加者の更新をゼロ化することで，標的参加者の個別更新値を抽出し，そのデータを再構築する攻撃を示している．

攻撃検出法として**モデル不一致検出**と**モデル整合性検証**が提案されているが，ともに限界がある．モデル不一致検出 [9] は，全参加者が同一のモデルをサーバから受信したか確認するが，全員に同一の改竄モデルが配布される一部の攻撃 [7], [10] に対しては攻撃検出が機能しない．モデル整合性検証 [11] は暗号学的手法により改竄の検出を試みるが，その際にかかる追加の計算・通信コストが大きい．

本稿では，データ再構築攻撃を防ぐ新たな枠組みとして Per-element SecAgg を提案する．Per-element SecAgg は，従来の SecAgg の機能を強化し，集約更新ベクトルの各要素について，t 個以上の非ゼロ貢献がある場合のみその集約値を開示することで，個別値の露出を防ぎ，データ再構築攻撃を無効化する．

本稿の貢献は以下である．(i) t 個以上の非ゼロ貢献がある要素についての集約のみを開示する Per-element SecAgg を，既存 SecAgg の拡張法として提案する．これは既存 SecAgg [3], [4] で用いられる暗号技術のみに依存する．(ii) 軽量な SecAgg である Flamingo [4] を拡張し，Per-element SecAgg を実現するプロトコルを再設計する．(iii) プロトコルの性能評価を行い，追加オーバーヘッドが許容範囲であり，学習モデル性能への影響も最小限であることを示す．

2. 予備知識

本節では，本稿の理解に必要な暗号技術および Flamingo SecAgg プロトコルを説明する．表記は表 1 にまとめる．

2.1 暗号技術

2.1.1 PKI を用いた Diffie-Hellman 鍵交換

ユーザ i は秘密鍵 $a_i \in \{0,1\}^\kappa$ を生成し，対応する公開鍵 g^{a_i} を公開鍵基盤 (PKI) を通じて公開する．ここで，κ はセキュリティパラメータであり，g は素数位数の巡回群の生成元である．ユーザ i, j のペアは，自身の秘密鍵と相手の公開鍵を用いて共有秘密 $s_{i,j} = g^{a_i a_j}$ を確立する．

2.1.2 擬似乱数生成器 (PRG)

$\mathsf{PRG}(r)$ は, 乱数シード $r \in \{0,1\}^\kappa$ を長い擬似乱数列へ拡張する．使用する PRG は安全であり，シードが秘匿される限り，出力は真の乱数と計算的に区別できない．

2.1.3 擬似乱数関数 (PRF)

$\mathsf{PRF}(k, x)$ は，秘密鍵 $k \in \{0,1\}^\kappa$ を用いて，入力 x を擬似乱数値へ写像する．使用する PRF は安全であり，鍵が秘匿される限り，出力は真の乱数と計算的に区別できない．

2.1.4 Shamir の秘密分散

Shamir の (ℓ, L) 秘密分散は，秘密 s を以下の 2 つのアルゴリズムを用いて処理する．$\mathsf{SS.share}(s, \ell, L)$ は s を L 個のシェア $\{\langle s\rangle_1, \ldots, \langle s\rangle_L\}$ に分割し，$\mathsf{SS.recon}(\langle s\rangle_{l l\in\mathcal{L}}) \to s$ は $|\mathcal{L}| \geq \ell$ を満たす部分集合から s を復元する．秘密分散では，ℓ 個未満のシェアからは s の情報は得られない．

2.2 Flamingo SecAgg プロトコル

SecAgg は，参加者の個別入力を秘匿したままサーバにその集約を開示するプロトコルである．本稿では Flamingo [4] に着目する．Flamingo は, 従来の SecAgg [3] でボトルネックだったアンマスク処理を，一部のユーザ集合である復号者に委任することで効率化を実現している．

2.2.1 セットアップフェーズ

このフェーズは学習開始前に一度だけ実行される．全ユーザペア $i, j \in \mathcal{N}$ は PKI を用いた Diffie-Hellman 鍵交換を通じて，長期秘密 $s_{i,j}$ と共通鍵 $k_{i,j}$ を確立する．長期秘密は, 後のラウンドでペア毎のマスクを生成する際に用いる．また，信頼できる乱数値 R [12] を用いて，全ユーザ集合 $\mathcal{N}$ から復号者集合 $\mathcal{D}$ を選出する．

2.2.2 報告フェーズ

ラウンド τ において，各参加者 $i \in \mathcal{C}$ は R に基づいて，自身とのペア集合 $\mathcal{A}_i \subset \mathcal{C}$ を決定する．i は，全てのペア $j \in \mathcal{A}_i$ に対して，長期秘密から $\mathsf{PRF}(s_{i,j}, \tau)$ によりラウンド固有のペアシード $r_{i,j}$ を生成する．加えて，i はランダムに個別シード r_i を生成する．これらを用いて，i は自身のモデル更新ベクトル $\boldsymbol{x}_i$ をマスクする．

$$[\![\boldsymbol{x}_i]\!] = \boldsymbol{x}_i + \underbrace{\mathsf{PRG}(r_i)}_{\text{(個別マスク)}} + \underbrace{\sum_{j\in\mathcal{A}_i} \pm\mathsf{PRG}(r_{i,j})}_{\text{(ペアマスク)}} \tag{1}$$

各ペアマスクは $i < j$ のとき正, それ以外は負である．

個別マスクを後に除去するため，参加者 i は r_i を秘

密分散し，$\{\langle r_i\rangle_u\}_{u\in\mathcal{D}} \leftarrow \mathsf{SS.share}(r_i, \ell, |\mathcal{D}|)$ を得る．各シェア $\langle r_i\rangle_u$ は復号者 u との共通鍵 $k_{i,u}$ で暗号化し，$\{[\![\langle r_i\rangle_u]\!]_{k_{i,u}}\}_{u\in\mathcal{D}}$ を生成する．最終的に，i はマスク済み入力 $[\![\boldsymbol{x}_i]\!]$ と暗号化シェアをサーバに送信する．

2.2.3 アンマスクフェーズ

サーバは受信したマスク済みベクトルを集約する．本説明では簡単のため，参加者の途中離脱がないものと仮定する．このとき，ペアマスクは相殺され，$\sum_{i\in\mathcal{C}}[\![\boldsymbol{x}_i]\!] = \sum_{i\in\mathcal{C}}\boldsymbol{x}_i + \sum_{i\in\mathcal{C}}\mathsf{PRG}(r_i)$ が得られる．

このフェーズでは，個別マスクを除去して平文 $\sum_{i\in\mathcal{C}}\boldsymbol{x}_i$ を復元する．そのために，サーバはシードシェアの暗号文を対応する復号者に送信する．各復号者 $u\in\mathcal{D}$ は，参加者 $i\in\mathcal{C}$ のシードシェアの暗号文 $[\![\langle r_i\rangle_u]\!]_{k_{i,u}}$ を受信し，共通鍵 $k_{i,u}$ で復号し，平文をサーバに返送する．

サーバは，$|\mathcal{D}_1|\geq\ell$ を満たす復号者集合 $\mathcal{D}_1\subseteq\mathcal{D}$ からシェアを収集し，$r_i \leftarrow \mathsf{SS.recon}(\{\langle r_i\rangle_u\}_{u\in\mathcal{D}_1})$ によりシード r_i を再構築する．再構築したシードに PRG を適用して個別マスクを再生成し，集約ベクトルから $\sum_{i\in\mathcal{C}}\mathsf{PRG}(r_i)$ を減算することで，平文 $\sum_{i\in\mathcal{C}}\boldsymbol{x}_i$ を得る．

3. Per-element SecAgg

本節では，システム・脅威モデル，および設計目標を定義する．これらに基づき，データ再構築攻撃に対抗する新たな枠組みである Per-element SecAgg を説明する．

3.1 システムモデル

サーバとユーザ集合 $\mathcal{N}$ からなるスター型トポロジを考える．既存 SecAgg [4], [5] に従い，ユーザには参加者と復号者の 2 つの役割を考える．各ラウンド τ では，信頼できる乱数値に基づき参加者集合 $\mathcal{C}\subset\mathcal{N}$ が選出される．参加者 $i\in\mathcal{C}$ はサーバからグローバルモデルを受信し，ローカルデータで学習した後，更新ベクトル $\boldsymbol{x}_i\in\mathbb{R}^{|\mathcal{K}|}$ を返送する．復号者集合 $\mathcal{D}\subset\mathcal{N}$ も同じ乱数値から選ばれ，Flamingo と同様に SecAgg 処理を支援するが，モデル学習は行わない．サーバは SecAgg を介して集約結果 $\sum_{i\in\mathcal{C}}\boldsymbol{x}_i$ を得る．

連合学習では，ネットワーク不安定性により一部ユーザが途中離脱する可能性がある．本稿では，復号者の離脱率を δ_D とする．なお，参加者の離脱は提案手法に影響を与えないため，本稿では定義しない．

3.2 脅威モデル

サーバは悪意ある敵対者として振る舞い，SecAgg プロトコルから任意に逸脱可能とする．さらにサーバは，参加者の割合 η_C，復号者の割合 η_D と共謀可能とする．Flamingo と同様に $\delta_D+\eta_D<1/3$ が成立し，離脱復号者と共謀復号者の集合は互いに素とする．敵対者の目標は，既存攻撃 [6], [7], [8], [9], [10] により SecAgg 下で正直な参加者のデータセットを再構築することである．

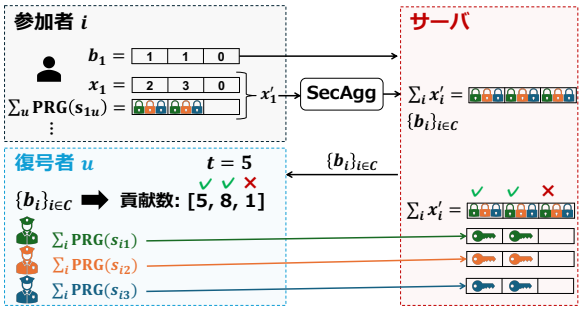

図 1 Per-element SecAgg フレームワークの概要．

3.3 設計目標

我々の主目標は，以下の性質 (P1)–(P4) を同時に満たすことでデータ再構築攻撃を防ぐことである．

(P1) 要素毎のしきい値集約． サーバは集約更新ベクトルについて，t 個以上の非ゼロ値が集約された要素についてのみその集約値を得る．

(P2) 悪意あるサーバに対するセキュリティ． サーバが悪意を持って振る舞っても，(P1) が保証される．具体的には，サーバによるメッセージの改竄を考慮する．

(P3) 離脱耐性． 最大 $\lfloor\delta_D\mathcal{D}\rfloor$ 人の復号者が離脱しても，プロトコルは中断せず，(P1) が保証される．参加者の離脱耐性も必要であるが，これは既存の SecAgg の処理 [3], [4] で実現可能なため，本稿では詳述しない．

(P4) 共謀耐性． サーバが最大 $\lfloor\eta_C\mathcal{C}\rfloor$ 人の参加者と最大 $\lfloor\eta_C\mathcal{D}\rfloor$ 人の復号者と共謀しても，(P1) が保証される．

一方，以下の性質は我々の目標ではない．

インデックス情報のプライバシ． 我々の提案手法は，各参加者が非ゼロ値を寄与した要素のインデックスをサーバに開示する．本研究では，この情報共有は脅威につながらないと考える．詳細な議論は 5.3 節で行う．

3.4 提案メカニズムの中核とその設計理由

2.1 節で説明した標準的な暗号技術のみを用いて，既存 SecAgg を Per-element SecAgg に拡張する手法を提案する．本手法の核心は，各要素で何人の参加者が非ゼロ値を寄与したかに基づいて，復号者がどの要素のアンマスクをサーバに許可するかを制御する点である．処理の流れは以下の通りである (図 1): 1) 各参加者は，復号者と共有するペアマスクを用いて更新ベクトルの非ゼロ要素のみをマスクする．2) マスク済みベクトルを収集後，サーバは各要素について非ゼロ値を寄与した参加者数をカウントし，この情報を復号者に報告する．3) しきい値 t を超える要素について，復号者は対応するマスクを返送し，サーバがその要素をアンマスクできるようにする．

この処理は，(P1) と (P2) を満たすよう設計されている．以下では，メカニズムを構成する 3 つの要素: 1) 要素毎の貢献数カウント，2) 要素毎のマスク，3) 要素毎のアンマスクについて概説する．詳細なプロトコルおよび (P3) と

(P4) への対応は，4 章と 5 章で説明する．

3.4.1 要素毎の貢献数カウント

(P1) 実現には，復号者は要素毎に非ゼロ貢献数が t 以上かを判定する必要がある．そのために，各参加者 $i \in \mathcal{C}$ は次のようなカウンタベクトル $\boldsymbol{b}_i \in \{0,1\}^{|\mathcal{K}|}$ を生成する:

$$\boldsymbol{b}_i[k] = \begin{cases} 1 & \text{if } \boldsymbol{x}_i[k] \neq 0 \\ 0 & \text{otherwise.} \end{cases}, \ k \in \mathcal{K} \qquad (2)$$

$\boldsymbol{b}_i$ は更新ベクトル $\boldsymbol{x}_i$ 内の非ゼロ値のインデックスを示す．参加者は $\boldsymbol{b}_i$ をサーバに送信し，サーバはこれを集めて $\{\boldsymbol{b}_i\}_{i\in\mathcal{C}}$ を復号者に送る．復号者は各要素について非ゼロ貢献数を計算し，アンマスクを行うかどうか判断する．

3.4.2 要素毎のマスク

(P1) 実現のため，サーバは $\{\boldsymbol{b}_i\}_{i\in\mathcal{C}}$ を復号者に転送する．しかし悪意あるサーバは，特定インデックス k について t 以上の貢献があったと $\{\boldsymbol{b}_i\}_{i\in\mathcal{C}}$ を偽造する可能性がある．この場合，正直な復号者は騙されてその要素についてのマスクを返し，(P2) が破られる．

この脅威を考慮し，提案手法では，カウンタベクトルの偽造によって不正なアンマスクが成功しないようにする．具体的には，各参加者は，ローカル更新 $\boldsymbol{x}_i$ の非ゼロ要素だけにマスキングを行う．これにより，SecAgg 後に得られる集約ベクトルで，各要素におけるマスク総和は，実際に非ゼロ値を貢献した参加者の数に依存する．サーバが偽の $\{\boldsymbol{b}_i\}_{i\in\mathcal{C}}$ を復号者に報告すると，返送されるアンマスク用のマスクはこのマスク総和と一致しないため，アンマスクに失敗する．これにより，悪意あるサーバによる偽造に対する耐性が実現できる．

3.4.3 要素毎のアンマスク

要素単位のアンマスク実現には，各復号者が要素単位でマスクを返す必要がある．Flamingo では，復号者がシードシェアを送ることで，サーバがマスクベクトル全体を PRG で再生成するが，この方針ではサーバが全要素を復号可能となるため，(P1) に違反する．

そこで提案手法では，復号者はシードやそのシェアではなく，マスクベクトル自体を返す．具体的にはしきい値 t を満たす要素に対してのみマスク値をサーバに開示する．これによりサーバに対してしきい値を満たす要素のみアンマスクを許可し，要素単位の秘匿性が保たれる．

4. プロトコル

Per-element SecAgg を実現する完全なプロトコルを提示する．これは，(P1) および (P2) のために 3 章で説明したメカニズムを組み込んで Flamingo を拡張し，復号者の離脱耐性（P3）のための追加フェーズを導入している．

4.1 (P1) および (P2) をカバーするプロトコルフェーズ

本節では，(P1) および (P2) を達成する報告フェーズとアンマスクフェーズを説明する．図 2 の赤色部分に示す拡張部分に焦点を当てる．簡素性のため，参加者離脱の回復

セットアップ

- **ユーザ** $i \in \mathcal{N}$
  - 鍵ペア $(g^{a_i}, a_i), (g^{b_i}, b_i)$ 生成
  - 公開鍵 (g^{a_i}, g^{b_i}) を PKI へ登録
- **全員**
  - パラメータ $\kappa, R, t, \ell, \Delta_{\max}$ を設定
  - R で復号者集合 $\mathcal{D}$ を選出

報告フェーズ（ラウンド τ）

- **参加者** $i \in \mathcal{C}$
  - R でペア集合 $\mathcal{A}_i$ を選出
  - PKI から $g^{a_j}(j \in \mathcal{A}_i)$, g^{a_u}, $g^{b_u}(u \in \mathcal{D})$ を取得
  - $s_{i,j} \leftarrow g^{a_i a_j}(j \in \mathcal{A}_i)$, $s_{i,u} \leftarrow g^{a_i a_u}$, $k_{i,u} \leftarrow g^{b_i b_u}(u \in \mathcal{D})$
  - $r_{i,j} \leftarrow \mathsf{PRF}(s_{i,j}, \tau)(j \in \mathcal{A}_i)$, $r_{i,u} \leftarrow \mathsf{PRF}(s_{i,u}, \tau)(u \in \mathcal{D})$
  - $\boldsymbol{b}_i$ を導出，$\boldsymbol{x}'_i = \boldsymbol{x}_i + \boldsymbol{b}_i \odot \sum_{u\in\mathcal{D}} \mathsf{PRG}(r_{i,u})$
  - r_i を生成，$[\![\boldsymbol{x}'_i]\!] = \boldsymbol{x}'_i + \mathsf{PRG}(r_i) + \sum_{j\in\mathcal{A}_i} \pm\mathsf{PRG}(r_{i,j})$
  - $\{\langle r_i\rangle_u\}_{u\in\mathcal{D}} \leftarrow \mathsf{SS.share}(r_i, \ell, |\mathcal{D}|)$, $\{\langle r_{i,v}\rangle_u\}_{u,v\in\mathcal{D}} \leftarrow \mathsf{SS.share}(\{r_{i,v}\}_{v\in\mathcal{D}}, \ell, |\mathcal{D}|)$
  - $\langle r_i\rangle_u$, $\{\langle r_{i,v}\rangle_u\}$ を $k_{i,u}$ で暗号化 $(u \in \mathcal{D})$
  - サーバへ送信:
    $[\![\boldsymbol{x}_i]\!]$, $\boldsymbol{b}_i$, $\{[\![\langle r_i\rangle_u]\!]_{k_{i,u}}\}_{u\in\mathcal{D}}$, $\{\{[\![\langle r_{i,v}\rangle_u]\!]_{k_{i,u}}\}_{u,v\in\mathcal{D}}$
- **サーバ**
  - $\sum_i [\![\boldsymbol{x}'_i]\!]$ を集約
  - 復号者 $u \in \mathcal{D}$ に $\{\boldsymbol{b}_i\}_{i\in\mathcal{C}}$, $\{[\![\langle r_i\rangle_u]\!]_{k_{i,u}}\}_{i\in\mathcal{C}}$ を送信

アンマスクフェーズ（ラウンド τ）

- **復号者** u
  - PKI から g^{a_i}, g^{b_i} を取得
  - $s_{i,u} \leftarrow g^{a_i a_u}$, $k_{i,u} \leftarrow g^{b_i b_u}(i \in \mathcal{C})$
  - $r_{i,u} \leftarrow \mathsf{PRF}(s_{i,u}, \tau)$ $(i \in \mathcal{C})$
  - $\{\boldsymbol{b}_i\}_{i\in\mathcal{C}}$ から $\mathcal{C}[k]$ を構成
  - $\boldsymbol{emk}_u[k] = \sum_{i\in\mathcal{C}[k]} \mathsf{PRG}(r_{i,u})[k]$ $(|\mathcal{C}[k]| \geq t)$，他は $\perp$
  - $\{[\![\langle r_i\rangle_u]\!]_{k_{i,u}}\}_{i\in\mathcal{C}}$ を $\{k_{i,u}\}_{i\in\mathcal{C}}$ で復号
  - サーバに $\boldsymbol{emk}_u$, $\{\langle r_i\rangle_u\}_{i\in\mathcal{C}}$ を送信
- **サーバ**
  - $\mathcal{D}_1$ からメッセージ収集
  - $|\mathcal{D}_1| \geq \ell$ なら $\{r_i\}_{i\in\mathcal{C}} \leftarrow \mathsf{SS.recon}(\{\{\langle r_i\rangle_u\}_{u\in\mathcal{D}_1}\}_{i\in\mathcal{C}})$，他はアボート
  - $\sum_{u\in\mathcal{D}_1} \boldsymbol{emk}_u$ と $\sum_{i\in\mathcal{C}} \mathsf{PRG}(r_i)$ を $\sum_{i\in\mathcal{C}} [\![\boldsymbol{x}'_i]\!]$ から減算
  - $\mathcal{D}_1 \neq \mathcal{D}$ なら $\mathcal{V} = \mathcal{D} \setminus \mathcal{D}_1$ を導出，他はラウンド終了
  - $u \in \mathcal{D}_1$ に $\mathcal{V}$ と $\{[\![\langle r_{i,v}\rangle_u]\!]_{k_{i,u}}\}_{i\in\mathcal{C},v\in\mathcal{V}}$ を送信

復号者離脱回復フェーズ（ラウンド τ，$\mathcal{D}_1 \neq \mathcal{D}$ の場合のみ）

- **復号者** $u \in \mathcal{D}_1$
  - $|\mathcal{V}| \geq \Delta_{\max}$ の場合のみ進行，他はアボート
  - $\{[\![\langle r_{i,v}\rangle_u]\!]_{k_{i,u}}\}_{v\in\mathcal{V}}$ を $\{k_{i,u}\}_{i\in\mathcal{C}}$ で復号，サーバへ返送
- **サーバ**
  - $\mathcal{D}_2 \subseteq \mathcal{D}_1$ からメッセージ収集
  - $|\mathcal{D}_2| \geq \ell$ なら $\{r_{i,v}\}_{i\in\mathcal{C},v\in\mathcal{D}\setminus\mathcal{D}_1} \leftarrow \mathsf{SS.recon}(\{\langle r_{i,v}\rangle_u\}_{i\in\mathcal{C},v\in\mathcal{C}})$，他はアボート
  - $\sum_{i\in\mathcal{C}} \sum_{u\in\mathcal{D}\setminus\mathcal{D}_1} \boldsymbol{b}_i \odot \mathsf{PRG}(r_{i,u})$ を $\sum_{i\in\mathcal{C}} [\![\boldsymbol{x}'_i]\!] - \sum_{u\in\mathcal{D}_1} \boldsymbol{emk}_u$ から減算

図 2 Flamingo を Per-element SecAgg に拡張するプロトコル．赤色は Flamingo の拡張部分．参加者離脱は考慮しない．

処理はここでは省略するが，Flamingo の元の手法 [4] を用いて本拡張に影響を与えずに実現可能である．

4.1.1 セットアップフェーズ

本フェーズは Flamingo のセットアップフェーズ (2.2 節) と同一である．

4.1.2 報告フェーズ

Per-element SecAgg におけるマスキングのため，各参加者 $i \in \mathcal{C}$ は PKI から全復号者 $u \in \mathcal{D}$ の公開鍵を取得し，長期秘密 $s_{i,u} \leftarrow g^{a_i a_u}$ を導出する．次に i はラウンド固有のペアシード $r_{i,u} \leftarrow \mathsf{PRF}(s_{i,u}, \tau)$ を計算する．

i は更新 $\boldsymbol{x}_i$ から式 (2) に従いカウンタベクトル $\boldsymbol{b}_i$ を構築する．次に， 3.4.2 節で説明したように，$\boldsymbol{x}_i$ の非ゼロ要素のみをマスクする: $\boldsymbol{x}'_i = \boldsymbol{x}_i + \boldsymbol{b}_i \odot \sum_{u\in\mathcal{D}} \mathsf{PRG}(r_{i,u})$. ここで $\odot$ は要素毎の乗算を表す．マスクされたベクトル $\boldsymbol{x}'_i$ は，次に Flamingo のマスキングを経て，最終的なベクトル $[\![\boldsymbol{x}'_i]\!]$ となる．

復号者の離脱回復 (次節で説明) の準備として, i は各シード $r_{i,u}$ を全復号者間で秘密分散する: $\{\langle r_{i,v}\rangle_u\}_{u,v\in\mathcal{D}} \leftarrow \mathsf{SS.share}(\{r_{i,v}\}_{v\in\mathcal{D}}, \ell, |\mathcal{D}|)$．次に各シェア $\langle r_{i,v}\rangle_u$ を共通鍵 $k_{i,u}$ で暗号化し $[\![\langle r_{i,v}\rangle_u]\!]_{k_{i,u}}$ を生成する．最後に，i はサーバへ $[\![\boldsymbol{x}'_i]\!]$，$\boldsymbol{b}_i$，暗号化シェア $\{[\![\langle r_i\rangle_u]\!]_{k_{i,u}}\}_{u\in\mathcal{D}}$ と $\{[\![\langle r_{i,v}\rangle_u]\!]_{k_{i,u}}\}_{u,v\in\mathcal{D}}$ を送信する．

参加者から受信後，サーバはベクトルを集約する:

$$\sum_{i\in\mathcal{C}} [\![\boldsymbol{x}'_i]\!] = \sum_{i\in\mathcal{C}} \boldsymbol{x}_i + \sum_{i\in\mathcal{C}} \mathsf{PRG}(r_i) + \sum_{i\in\mathcal{C}}\sum_{u\in\mathcal{D}} \boldsymbol{b}_i \odot \mathsf{PRG}(r_{i,u}). \tag{3}$$

式 (3) のアンマスクのために, サーバは $\{[\![\langle r_i\rangle_u]\!]_{k_{i,u}}\}_{u\in\mathcal{D}}$ と $\{\boldsymbol{b}_i\}_{i\in\mathcal{C}}$ を各復号者 $u \in \mathcal{D}$ に転送する．これにより 3.4.1 節で説明した転送が完了し，復号者は要素毎の非ゼロ値貢献者数を計算できる．

4.1.3 アンマスクフェーズ

各復号者 $u \in \mathcal{D}$ はまず参加者との長期秘密から，ラウンド固有シード $\{r_{i,u}\}_{i\in\mathcal{C}}$ を導出する．次にサーバから受信した $\{\boldsymbol{b}_i\}_{i\in\mathcal{C}}$ を用いて，u は各インデックス $k \in \mathcal{K}$ の非ゼロ貢献者の集合を $\mathcal{C}[k] = \{i \in \mathcal{C} \mid \boldsymbol{b}_i[k] = 1\}$ として構築する．貢献者数 $|\mathcal{C}[k]|$ に基づき，u はしきい値 t を満たす要素を判定し，要素毎のマスクベクトル $\boldsymbol{emk}_u$ を生成する:

$$\boldsymbol{emk}_u[k] = \begin{cases} \sum_{i\in\mathcal{C}[k]} \mathsf{PRG}(r_{i,u})[k] & \text{if } |\mathcal{C}[k]| \geq t \\ \perp & \text{otherwise.} \end{cases} \tag{4}$$

ここで $\perp$ は無効値，$\mathsf{PRG}(r_{i,u})[k]$ は乱数マスクベクトルの k 番目要素を表す．このように u は， 3.4.3 節で述べたように，アンマスクの対象となる要素を明示的に制御する．最後に復号者は $\boldsymbol{emk}_u$ と復号済みシードシェア $\{\langle r_i\rangle_u\}_{i\in\mathcal{C}}$ をサーバに返す．

全復号者からの受信後，サーバは個別シードを再構築し $\sum_{i\in\mathcal{C}} \mathsf{PRG}(r_i)$ を再生成する．次に要素毎のマスクの合計 $\sum_{u\in\mathcal{D}} \boldsymbol{emk}_u$ を使用して，サーバはアンマスクを実行する:

$$\begin{aligned} \boldsymbol{y} &= \sum_{i\in\mathcal{C}} [\![\boldsymbol{x}'_i]\!] - \sum_{i\in\mathcal{C}} \mathsf{PRG}(r_i) - \sum_{u\in\mathcal{D}} \boldsymbol{emk}_u \\ &= \begin{cases} \sum_{i\in\mathcal{C}} \boldsymbol{x}_i[k] & \text{if } |\mathcal{C}[k]| \geq t \\ \perp & \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

これにより，t 人以上の非ゼロ貢献者がいる要素のみアンマスクされ，(P1) が満たされる．

4.2 (P3) をカバーするプロトコルフェーズ

4.2.1 動機

アンマスクフェーズでマスクを返送する復号者集合を $\mathcal{D}_1 \subseteq \mathcal{D}$ とする．離脱が生じて $\mathcal{D}_1 \neq \mathcal{D}$ となれば，対応するマスクが欠落し，式 (5) のアンマスクに失敗する．よって， $\mathcal{D}_1 \neq \mathcal{D}$ の場合のみ発動する離脱回復フェーズが必要である．

4.2.2 方針

参加者は報告フェーズで事前に全シードの暗号化シェアをサーバに送信する．復号者の離脱が発生したら，サーバは離脱リスト $\mathcal{V} = \mathcal{D} \setminus \mathcal{D}_1$ と対応する暗号文シェアを生存復号者 $\mathcal{D}_1$ に送信し，復号を依頼する．すなわち，サーバは離脱復号者分についてのみ，そのシードを取得し，欠落したマスクをサーバで再生成する．このとき，アンマスクは依然として $\mathcal{D}_1$ 内の正直な復号者の要素毎マスク $\boldsymbol{emk}_u$ に依存するため， (P1) は維持される．

4.2.3 (P2) 違反のリスクとその解決策

悪意あるサーバは，正直かつ実際には生存している復号者を虚偽の $\mathcal{V}$ に含め，全てのシードを取得することで，全要素についてのアンマスクを試みる恐れがある．この場合，(P1) が破綻する．この対策として，許容される離脱数の上限 $\Delta_{\max}$ を導入する．もし $|\mathcal{V}| > \Delta_{\max}$ なら，正直な復号者はプロトコルをアボートする．5.2 節で証明するが，$\Delta_{\max} = \lceil \ell/2 \rceil$ と設定すれば，復号者の離脱回復を阻害することなく (P2) を保証できる．

4.2.4 手順

アンマスクフェーズで復号者の離脱 (すなわち $\mathcal{D}_1 \neq \mathcal{D}$) を検出すると，サーバは $\mathcal{V} = \mathcal{D} \setminus \mathcal{D}_1$ を構築し，これと暗号化シードシェア $\{[\![\langle r_{i,v}\rangle_u]\!]_{k_{i,u}}\}_{i\in\mathcal{C},v\in\mathcal{V}}$ を生存復号者 $u \in \mathcal{D}_1$ に送信する．復号者 $u \in \mathcal{D}_1$ はまず $|\mathcal{V}| \leq \Delta_{\max}$ を検証した上で，共通鍵 $\{k_{i,u}\}_{i\in\mathcal{C}}$ を用いて暗号文を復号し，シェア平文をサーバに返す．サーバは少なくとも ℓ 人の復号者から応答を収集し，秘密分散により欠落マスクのシード $\{r_{i,v}\}_{i\in\mathcal{C},v\in\mathcal{V}}$ を再構築し，カウンタベクトル $\{\boldsymbol{b}_i\}_{i\in\mathcal{C}}$ と併せてアンマスクを完了する:

$$\begin{aligned} \boldsymbol{y} = \sum_{i\in\mathcal{C}} [\![\boldsymbol{x}'_i]\!] &- \sum_{i\in\mathcal{C}} \mathsf{PRG}(r_i) \\ &- \sum_{u\in\mathcal{D}_1} \boldsymbol{emk}_u - \sum_{v\in\mathcal{D}\setminus\mathcal{D}_1} \boldsymbol{b}_i \odot \mathsf{PRG}(r_{i,v}). \end{aligned} \tag{6}$$

以上により，正しい集約値 $\boldsymbol{y}$ が得られ，(P3) が保証される．

表 2　プロトコルフェーズ毎の計算・通信コスト

	参加者+復号者	サーバ
	計算コスト	
報告	$\mathcal{O}(DK' + AK + D^3)$	$\mathcal{O}(CK)$
アンマスク	$\mathcal{O}(CK')$	$\mathcal{O}(DK' + CD^2 + CK)$
離脱回復	$\mathcal{O}(CV)$	$\mathcal{O}(V(CD^2 + K'))$
	通信コスト	
報告	$\mathcal{O}(K + \alpha K' + D^2)$	$\mathcal{O}(C(D^2 + DK'\alpha + K))$
アンマスク	$\mathcal{O}(CK'\alpha)$	$\mathcal{O}(D(K' + CV))$
離脱回復	$\mathcal{O}(CV)$	$\mathcal{O}(CDV)$

表記: C: 参加者数，D: 復号者数，A: 各参加者のペア集合サイズ，V: 離脱復号者数，K: モデル更新ベクトル $\boldsymbol{x}_i$ の次元，K': 追加マスク対象となるベクトルの次元，α: 更新スパース率.

5. コスト・セキュリティ・プライバシ分析

5.1 計算・通信コスト分析

我々のプロトコルの計算・通信コストを表 2 に示す．計算コストは次の操作を $\mathcal{O}(1)$ と定義する: 1 回の PRF 評価、ベクトル要素 1 つに対する PRG 出力，1 回の共通鍵暗号化/復号化，およびベクトル要素 1 つの加算/減算である．通信コストについては，$\mathcal{O}(1)$ をベクトル要素 1 個とシェア 1 個の転送として定義する.

5.2 セキュリティ分析

本節では，我々のプロトコルが (P1)–(P4) を満たすことを示す．まず，共謀耐性 (P4) のためのパラメータ設定を述べ，続いて悪意あるサーバ，共謀参加者・復号者，復号者の離脱が存在する場合の安全性を示す.

5.2.1 (P4) のためのパラメータ設定

式 (4) に示すように，復号者は要素の非ゼロ貢献者数がしきい値に達したときのみマスクを返す．しかし，共謀参加者は $\boldsymbol{b}_i$ を偽装し，実際より多くの貢献があるように見せかけ，しきい値を不正に満たす可能性がある．このため，復号者がマスク返送の判断に用いるしきい値を t から t' に再定義する．直感的には，共謀者が水増しできる人数分を上乗せする.

定理 1. サーバが最大で $\lfloor \eta_C |\mathcal{C}| \rfloor$ 人の参加者と共謀する場合，復号者の判断しきい値を $t' = \lfloor \eta_C |\mathcal{C}| \rfloor + t$ と再定義すれば，少なくとも t 人の正直な参加者が貢献しない限り要素はアンマスクされない.

次に，共謀復号者に対する耐性を実現するための ℓ と $\Delta_{\max}$ の設定を述べる.

定理 2. サーバが最大 $\lfloor \eta_D |\mathcal{D}| \rfloor$ 人の復号者と共謀し，さらに最大 $\lfloor \delta_D |\mathcal{D}| \rfloor$ 人の正直な復号者が離脱する場合を考える．ただし $\delta_D + \eta_D < 1/3$ とする．このとき $\ell = \lfloor 2|\mathcal{D}|/3 \rfloor + 1$, $\Delta_{\max} = \lceil \ell/2 \rceil$ と設定すれば，離脱復号者のシードをすべて復元可能であり (回復機能)，サーバが正直な復号者全員のシードを復元することはできない (セキュリティ機能).

略証. 各生存復号者は離脱復号者 v ごとに $|\mathcal{C}|$ 個のシェアを返し，これを 1 単位とする．サーバが受信可能な単位数は $A = \Delta_{\max} \cdot \lfloor (1 - \delta_D - \eta_D) |\mathcal{D}| \rfloor$ である.

回復機能には $B = \lfloor \delta_D |\mathcal{D}| \rfloor \cdot (\ell - \lfloor \eta_D |\mathcal{D}| \rfloor)$ 個の単位が必要であり，セキュリティ機能には $C = \lfloor (1 - \eta_D) |\mathcal{D}| \rfloor \cdot (\ell - \lfloor \eta_D |\mathcal{D}| \rfloor)$ 未満である必要がある．したがって $B \le A < C$ を満たすように $\ell, \Delta_{\max}$ を設定する．最悪ケース (回復は $\delta_D \to 1/3$，セキュリティは $\eta_D \to 1/3$) を考慮すると，最適設定は $\ell = \lfloor 2|\mathcal{D}|/3 \rfloor + 1$，$\Delta_{\max} = \lceil \ell/2 \rceil$ となる． □

5.2.2 要素ごとのしきい値集約

以上を踏まえ，我々のプロトコルが要素毎のしきい値集約を保証することを示す.

定理 3. パラメータを $t' = \lfloor \eta_C |\mathcal{C}| \rfloor + t$, $\ell = \lceil 2|\mathcal{D}|/3 \rceil + 1$, $\Delta_{\max} = \lfloor \ell/2 \rfloor$ と設定する．このとき，サーバが最大で $\lfloor \eta_C |\mathcal{C}| \rfloor$ 人の参加者および $\lfloor \eta_D |\mathcal{D}| \rfloor$ 人の復号者と共謀し，さらに最大で $\lfloor \delta_D |\mathcal{D}| \rfloor$ 人の復号者が離脱しても，サーバは少なくとも t の正直な参加者が貢献した要素の合計のみを学習でき，貢献が不足する要素は秘匿される.

略証. ハイブリッド論法による厳密な証明は省略する．Flamingo プロトコルのセキュリティにより，サーバは常にマスクされた合計のみを得る．貢献者数が t 以上の場合、復号者は対応するマスクを返し，サーバは正しい合計を復元できる．反対に、貢献者が不足する場合は少なくとも 1 人の正直な復号者がマスクを保留し，PRG 出力の予測不可能性により計算上乱数と識別できない．定理 1 と定理 2 によって，サーバはカウントの偽造や離脱操作を通じてこのマスクにアクセスすることもできない．したがって、要素は必要な閾値を満たす場合にのみ公開される． □

5.3 プライバシ分析

提案手法は，カウンタベクトル $\boldsymbol{b}_i$ を介して，各参加者が非ゼロ値を持つ要素のインデックスをサーバと復号者に開示する．本節は，これによるプライバシ影響を議論する.

まず, データ再構築の懸念が考えられる．Us Sami ら [13] は，SecAgg において非ゼロ更新のインデックスを知ることで，サーバが線形式を解き参加者のデータを再構築できると指摘している．しかし，この攻撃は同一参加者に対し固定グローバルモデルを数百ラウンド送信することを前提とし，各ラウンドで参加者がランダムに選択される我々のシステムモデル (3.1 節) では現実的ではない.

次に，属性推論のリスクがある．Pasquini ら [9] は，インデックスの観察により，サーバが参加者データの属性を推定できることを示している．提案手法は，3.3 節で述べたように再構築攻撃の防御を目的とし，属性推論までは扱わない．この課題への対応は今後の検討に委ねる.

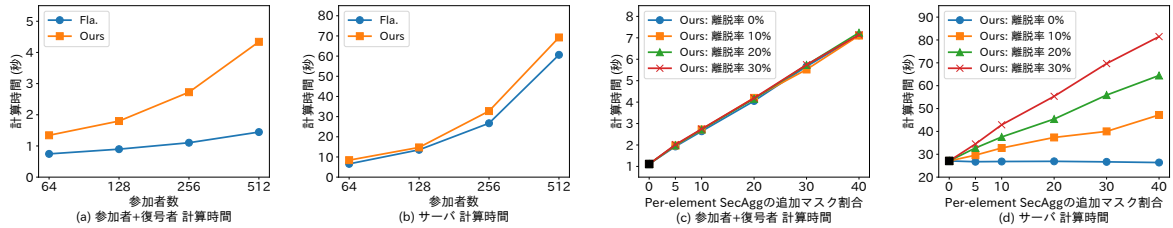

図 3 参加者数と追加マスク割合変更時における計算時間 (モデル更新ベクトル次元: 5M, 復号者数: 40)

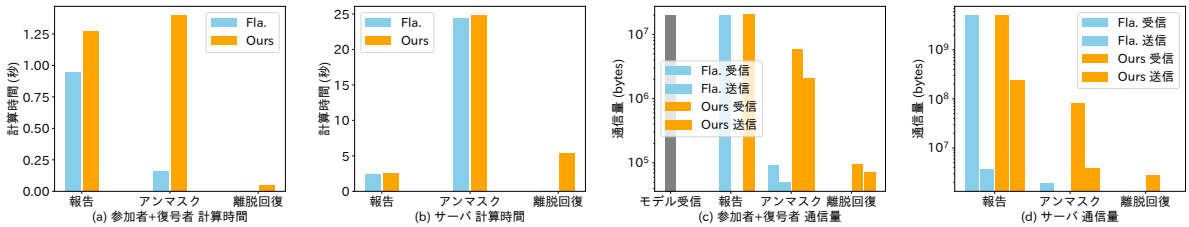

図 4 各プロトコルフェーズにおけるオーバーヘッドの内訳 (モデル更新ベクトル次元: 5M, 参加者数: 256, 復号者数: 40, 復号者の離脱率: 10%)

6. 評価

我々は, 提案プロトコルを Flamingo と比較し, 追加オーバーヘッドを評価する. また, Per-element SecAgg がモデル性能に与える影響も検証する.

6.1 実験セットアップ

6.1.1 実装

提案プロトコルを ABIDES シミュレータ [14] 上に実装した. このシミュレータは既存の SecAgg プロトコルの評価にも用いられている [4], [5].

6.1.2 パラメータ設定

現実的な連合学習と脅威シナリオを想定し, 以下の通りパラメータを設定する.

追加マスク率. データ再構築攻撃では出力層を含む線形層部分の更新のみが再構築のための情報として狙われるため [7], Per-element SecAgg のマスキングを更新ベクトルのうちその一部分のみに適用することとする. この場合, マスク対象は一般に更新全体の 10%未満に収まるが, 線形層を挿入して再構築する攻撃 [10] のように, より高い割合が必要となる場合もある. 本稿では, 更新ベクトル次元を 5M に固定し, マスク率を 5%から 40%まで変化させる.

更新ベクトルのスパース性. 再構築攻撃を防ぐため, 更新ベクトル内で絶対値の小さな要素 (しきい値 λ 未満) はゼロとして扱う. これは連合学習におけるしきい値ベースのスパース化手法 [15] に一致する. [15] によれば, λ を 10^{-3}〜10^{-2} に設定すると 99%以上がゼロ化される. 本評価では保守的に 95%のスパース性を仮定する.

6.2 計算時間と通信オーバーヘッド

6.2.1 追加マスク率が 10%の場合

追加マスク率を 10%に固定して評価する. 図 3a,b より, 参加者数 256 人時の追加計算時間はユーザ側 1.6 秒, サーバ側 6.0 秒, 合計 7.6 秒に過ぎない. 図 4 から, ユーザ側の大半 (75%) は復号者によるアンマスク処理に起因すると分かる. 復号者は参加者と同じユーザ集合に属するが, 参加者は学習やモデル送受信といったより重い処理を担うため, 復号者のこの追加負荷は十分許容できる.

通信では, 図 4c,d に示す通り, ユーザ側 1.21 倍, サーバ側 1.07 倍の増加に留まる. ユーザ側の増加の大部分 (96%) は復号者のカウンタベクトル受信とマスク送信によるが, 参加者によるモデル受信や更新送信と比べれば支配的ではない. 以上より, 計算・通信の両方で, 追加オーバーヘッドは主要処理に比べ小さく, 実用上許容範囲に収まる.

6.2.2 追加マスク率を変化させた場合

図 3c,d は追加マスク率および復号者離脱率を変化させた場合の計算時間を示す. 0%は Flamingo に対応する. ユーザ側では計算時間はマスク率にほぼ比例して増加するが, 離脱率の影響は小さい. 最大はマスク率 40%時で Flamingo の 6.4 倍である. サーバ側でもマスク率に比例して増加し, 離脱率が高いほど傾きが大きいが, 最大でも 40%・離脱率 30%時で Flamingo の 2.9 倍に留まる. 全体として, 支配的な処理は既存の学習やモデル通信であり, 提案手法による追加部分は相対的に小さいため, オーバーヘッドは十分に許容可能である.

6.3 モデル性能への影響

Per-element SecAgg では, 非ゼロ貢献が t 未満の要素についてモデルを更新できないため, モデルの性能に影響が

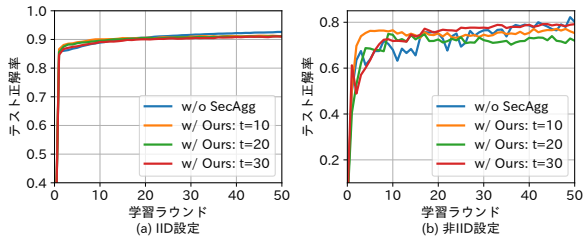

図 5 Per-element SecAgg 下におけるテスト正解率の遷移 (参加者数: 100，最適化アルゴリズム: FedAvg，データセット: MNIST，各参加者におけるローカル訓練エポック数: 5)

及ぶ懸念がある．本節ではその影響を検証する．

IID 設定 (図 5a) では，Per-element SecAgg は SecAgg 非使用時と同等の収束速度と正解率を示し，しきい値を大きくしても差はほとんど生じない．非 IID 設定 (図 5b) では，しきい値が大きい場合に正解率がやや低下する．

この違いは非ゼロインデックスの重複度に起因する．IID 環境では更新が重なりやすくしきい値を満たしやすいのに対し，非 IID 環境では重複が減少し [16]，一部の更新が反映されにくくなるためである．それでもなお，全体として正解率の低下幅は限定的であり，Per-element SecAgg の導入によるモデル性能への影響は最小限に留まる．

7. 関連研究

本章では，Per-element SecAgg で防御可能な，モデル更新のスパース性を悪用するデータ再構築攻撃をまとめる．

代表的な手法は，ReLU 活性化が負の入力に対してゼロを出力する性質を利用するものである．悪意あるサーバはモデル重みを改変し，特定サンプルのみが非ゼロ出力を生じるよう誘導することで，対応する更新から勾配情報を抽出できる [8], [9]．さらに，再構築精度を高めるため，モデルパラメータだけでなくアーキテクチャを操作する攻撃も提案されている．例えば，Fowl ら [6] や Zhao ら [10] は，グローバルモデルに ReLU 付き線形層を挿入し，対応する更新からデータを直接抽出可能であることを示した．また，Wen ら [7] は，重みやバイアスを精緻に調整することで特定クラスや特徴の勾配のみを増幅し，クライアント単位ではなく特徴単位での更新分離を実現している．この場合，全クライアントに同一の改竄モデルを配布してもデータ抽出が成立し，SecAgg を効果的に回避できる．

8. おわりに

本稿では，連合学習における SecAgg を拡張した Per-element SecAgg を提案した．本手法は，少なくとも t 人の参加者が非ゼロ要素を貢献した場合にのみ集約値を公開するものであり，Flamingo に組み込むことで新たな暗号技術を導入することなく実現できる．評価の結果，追加オーバーヘッドは小さく，モデル性能への影響も限定的であり，実用性が確認された．

謝辞 本研究は，科研費 24K22295 によるものである．

参考文献

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

[2] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM*, 2019.

[3] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *ACM CCS*, 2017.

[4] Yiping Ma, Jess Woods, Sebastian Angel, Antigoni Polychroniadou, and Tal Rabin. Flamingo: Multi-round single-server secure aggregation with applications to private federated learning. In *IEEE S&P*, 2023.

[5] Harish Karthikeyan and Antigoni Polychroniadou. OPA: One-shot private aggregation with single client interaction and its applications to federated learning. In *Crypto*, 2025.

[6] Liam H Fowl, Jonas Geiping, Wojciech Czaja, Micah Goldblum, and Tom Goldstein. Robbing the fed: Directly obtaining private data in federated learning with modified models. In *ICLR*, 2022.

[7] Yuxin Wen, Jonas A Geiping, Liam Fowl, Micah Goldblum, and Tom Goldstein. Fishing for user data in large-batch federated learning via gradient magnification. In *ICML*, Vol. 162, 2022.

[8] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. When the curious abandon honesty: Federated learning is not private. In *IEEE EuroS&P*, 2023.

[9] Dario Pasquini, Danilo Francati, and Giuseppe Ateniese. Eluding secure aggregation in federated learning via model inconsistency. In *ACM CCS*, 2022.

[10] Joshua C Zhao, Atul Sharma, Ahmed Roushdy Elkordy, Yahya H Ezzeldin, Salman Avestimehr, and Saurabh Bagchi. Loki: Large-scale data reconstruction attack against federated learning through model manipulation. In *IEEE S&P*, 2024.

[11] Yin Zhu, Junqing Gong, Kai Zhang, and Haifeng Qian. Malicious-resistant non-interactive verifiable aggregation for federated learning. *IEEE Trans. Dependable Secure Comput.*, Vol. 21, No. 6, 2024.

[12] Cloudflare randomness beacon. `https://developers.cloudflare.com/randomness-beacon/`.

[13] Hasin Us Sami and Başak Güler. Secure gradient aggregation with sparsification for resource-limited federated learning. *IEEE Trans. Commun.*, Vol. 72, No. 11, 2024.

[14] David Byrd, Maria Hybinette, and Tucker Hybinette Balch. ABIDES: Towards high-fidelity multi-agent market simulation. In *ACM SIGSIM-PADS*, 2020.

[15] Rongwei Lu, Yutong Jiang, Jinrui Zhang, Chunyang Li, Yifei Zhu, Bin Chen, and Zhi Wang. γ-FedHT: Stepsize-aware hard-threshold gradient compression in federated learning. In *IEEE INFOCOM*, 2025.

[16] Xinchi Qiu, Javier Fernandez-Marques, Pedro P B Gusmao, Yan Gao, Titouan Parcollet, and Nicholas Donald Lane. ZeroFL: Efficient on-device training for federated learning with local sparsity. In *ICLR*, 2022.