ドロップアウト耐性のある連合学習における計算と通信コストのトレードオフ調整

増田 大輝[1,a)]　北 健太朗[1]　小泉 佑揮[1]　武政 淳二[1]　長谷川 亨[1]

概要：連合学習は教師データのプライバシーを保護するための有望な分散機械学習システムである．しかし，ユーザのローカルモデルからデータが漏洩するリスクがある．このリスクに対処するため，ローカルモデルのマスクによるセキュアアグリゲーションプロトコルが提案されているが，既存のプロトコルは，ドロップアウトユーザの対策において計算と通信コストの間にトレードオフがある．ナイーブなプロトコルは，マスクの代わりに鍵を交換するため通信コストが低いが，サーバによる鍵からのマスクの復元に高い計算コストがかかる．最先端のプロトコルは，マスクを交換することでサーバの計算コストを削減するが，ユーザによるマスク交換に高い通信コストがかかる．本研究では，両方のプロトコルの強みを生かし，計算と通信コストのトレードオフを調整するセキュアアグリゲーションプロトコルを提案する．

Balancing Computation and Communication for Dropout-tolerant Federated Learning

Hiroki Masuda[1,a)]　Kentaro Kita[1]　Yuki Koizumi[1]　Junji Takemasa[1]　Toru Hasegawa[1]

Abstract: Federated learning is a promising distributed learning system that protects the privacy of training data. However, there is still a risk of data leakage from users' local models. To address this risk, secure aggregation using local model masking has been proposed, but existing secure aggregation protocols face a trade-off between computation and communication costs for handling dropout users. A naive protocol achieves a low communication cost by exchanging keys instead of masks, but it requires a high computation cost for the server to reconstruct masks from these keys. On the other hand, a state-of-the-art protocol reduces the server's computation cost by exchanging the masks directly, but this mask-exchange results in a high communication cost. This paper proposes a secure aggregation protocol that balances the trade-off between the computation and communication costs by complementing the strengths of both protocols.

1. はじめに

医療や金融など，データ漏洩が問題視される分野への機械学習システムの導入に向け，教師データを他者へ公開することなく機械学習モデルを作製する連合学習 [1] が注目されている．連合学習では，教師データを所持するデバイス (ユーザと呼ぶ) とサーバが，ラウンドと呼ばれる以下のプロセスを繰り返すことで機械学習モデル（グローバルモデルと呼ぶ）を学習する．各ラウンドにおいて，各ユーザは自身の教師データでグローバルモデルを学習し，学習したモデル（ローカルモデルと呼ぶ）をサーバへ送信する．サーバは受信したローカルモデルを集約（加算）し，ローカルモデルの和を用いてグローバルモデルを更新する．

しかし，連合学習はローカルモデルから教師データを推測する攻撃に対して脆弱である [2]．この問題に対処するため，ユーザのローカルモデルをサーバや他のユーザと共有せずに集約するセキュアアグリゲーションプロトコルが提案されている [3], [4], [7], [8], [9], [10], [11], [12]．これらのプロトコルはプライバシー保護とドロップアウト耐性を目標としている．プライバシー保護はオネストなユーザのローカルモデルがセミオネストなサーバとセミオネストな

[1] 大阪大学 大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
[a)] h-masuda@ist.osaka-u.ac.jp

プロトコル	計算コスト		通信コスト	
	ユーザ	サーバ	ユーザ	サーバ
SecAgg [3]	$\mathcal{O}(nm)$	$\mathcal{O}((n-r)m+r(n-r)m)$	$\mathcal{O}(m)$	$\mathcal{O}(nm)$
LightSecAgg [4]	$\mathcal{O}(nm)$	$\mathcal{O}(m)$	$\mathcal{O}(nm)$	$\mathcal{O}(n^2m)$
BalancedSecAgg	$\mathcal{O}(nm)$	$\mathcal{O}(rm)$	$\mathcal{O}(rm)$	$\mathcal{O}(rnm)$

表 1: SecAgg，LightSecAgg および BalancedSecAgg のコスト分析．n と r はそれぞれ，全ユーザと離脱ユーザ数を表す．

Table 1 A cost analysis of SecAgg, LightSecAgg, and BalancedSecAgg. n is the number of all users and r is the number of dropout users.

ユーザに漏洩しないこと，ドロップアウト耐性はドロップアウトユーザ（離脱ユーザと呼ぶ）が存在してもローカルモデルを集約できることである．

プライバシー保護とドロップアウト耐性を満たすための主なアイデアは 2 つある．第 1 に，各ユーザはローカルモデルをランダムベクトル（ランダムマスクと呼ぶ）でマスクしてからサーバへ送信する．第 2 に，サーバは，受信したドロップアウトしていないユーザ（生存ユーザと呼ぶ）のマスクされたローカルモデルを集約し，生存ユーザや離脱ユーザのランダムマスクを打ち消して，生存ユーザのローカルモデルの和を算出する．重要な準備として，事前にユーザ間でマスクなどの情報を交換する．

情報の交換には代表的な 2 つの手法があるが，**表 1** に示すように，計算と通信コストのうち，一方の効率は良いがもう一方の効率は悪いことが課題である．ここで，計算コストの単位はモデルやマスクなどの m 次元ベクトルの要素を 1 つ生成すること，通信コストの単位は m 次元ベクトルの要素を 1 つ転送することと定義する．

ナイーブな手法である SecAgg [3] は，通信コストは低いが計算コストが高い．マスクの代わりにランダム鍵やランダム鍵を冗長化して交換するため，ユーザがマスクを転送する回数が少ない．一方，サーバがランダム鍵からランダムマスクを復元する回数が多い．具体的には，サーバは各生存ユーザに対して 1 回，各離脱ユーザに対して $n-r$ 回マスクを復元するため，計算コストは $\mathcal{O}((n-r)m+r(n-r)m)$ となる．ここで，n は全ユーザ数，r は離脱ユーザ数である．最先端の手法である LightSecAgg [4] は，計算コストは低いが通信コストは高い．マスクそのものを冗長化して交換するため，サーバがマスクを復元する回数が少ない．一方，ユーザが冗長ランダムマスクを転送する回数が多い．具体的には，各ユーザは他のすべてのユーザへ冗長ランダムマスクを送信するため， 通信コストは $\mathcal{O}(nm)$ である．

本研究では，SecAgg の計算コストと LightSecAgg の通信コストのトレードオフを調整する手法を提案する．主なアイデアは，SecAgg と LightSecAgg の両方の強みを生かすことである．全体で n 台のユーザがいるとしたとき，$t+1$ 台のグループと $r=n-(t+1)$ 台のグループに分割する．各ユーザは，$t+1$ 個のランダムマスクを生成し，これらのランダムマスクからリードソロモン消失訂正符号 [17], [18] を用いて r 個の冗長ランダムマスクを生成する．その後，各ユーザはすべてのランダムマスクと冗長ランダムマスクを自身のローカルモデルに加えてサーバに送信する．生存ユーザのローカルモデルの和を計算するため，生存ユーザのマスクされたローカルモデルの和から減算する必要がある生存ユーザのランダムマスクと冗長ランダムマスクを以下のように生成する．まず，ユーザ間でランダムマスクと冗長ランダムマスクを交換し，集約する．この際，各ユーザの $t+1$ 個のランダムマスクの交換にはランダム鍵を使用する．次に，サーバは生存ユーザの集約されたマスクを収集し，これらのマスクを用いて離脱ユーザが送信しなかった集約されたマスクを復元する．この復元は，離脱ユーザの数が r 以下であれば可能である．提案手法では，各ユーザの通信コストは LightSecAgg と比べて $\mathcal{O}(rm)$ に，サーバの計算コストは SecAgg と比べて $\mathcal{O}(rm)$ に削減される．提案手法を BalancedSecAgg と呼ぶ．

本研究の貢献は以下の通りである：

- 計算と通信コストのトレードオフを調整するセキュアアグリゲーションプロトコルを設計する．
- 提案手法が SecAgg や LightSecAgg と同じ安全性の強度である $t \leq n-r-1$ の条件下で，プライバシー保護とドロップアウト耐性をもつことを証明する．ここで，t はセミオネストなユーザ数である．

本章の構成は以下の通りである．2 章では，関連研究について説明する．3 章では，問題設定について説明する．4 章では，提案手法である BalancedSecAgg を説明し．5 章では，その安全性を証明する．最後に 6 章で結論を述べる．

2. 関連研究

2.1 SecAgg

ランダム鍵を冗長化して交換する SecAgg は．マスクの交換が不要なため通信コストが低いが，ランダム鍵からマスクを復元する回数が多いため計算コストが高い．

各ユーザは他のユーザとのペア毎に共有する共有ランダム鍵と自身だけが持つ自己ランダム鍵を生成し，それらから共有ランダムマスクと自己ランダムマスクを生成してローカルモデルをマスクし，サーバへ送信する．このマスクはサーバの集約時に共有ランダムマスクが打ち消されるように設定しているため，集約後には生存ユーザの自己ラ

表 2: 記号の説明
Table 2 Description of Symbols

記号	説明
n	全ユーザ数
t	セミオネストなユーザ数
r	離脱ユーザ数
$\mathbf{w}$	グローバルモデル
$\mathbf{w}_i$	ユーザ i のローカルモデル
$\widetilde{\mathbf{w}}_i$	ユーザ i のマスクされたローカルモデル
$\mathcal{U}$	全ユーザの集合
$\mathcal{T}$	セミオネストなユーザの集合
$\mathcal{D}$	マスクされたローカルモデルをサーバへ送信しない離脱ユーザの集合
$\mathbf{PRG}(s_{i,j})$	ユーザ i の，ユーザ j 宛のランダムマスク
$\mathbf{d}_{i,j}$	ユーザ i の，ユーザ j 宛の冗長ランダムマスク

ンダムマスクと離脱ユーザの共有ランダムマスクが残る．残るランダムベクトルを打ち消すため，シャミアの秘密分散 [5] によって各ユーザは共有ランダム鍵と自己ランダム鍵を復元できる情報を事前に秘密分散する．サーバはこれらのシェアを収集し，残るランダムベクトルを復元して打ち消して，生存ユーザのローカルモデルの和を算出する．

各ユーザはマスクされたローカルモデルのみを m 次元ベクトルとして転送するため，通信コストは $\mathcal{O}(m)$ となる．一方，サーバは各生存ユーザの自己ランダムマスクと各離脱ユーザの $n-r$ 個の共有ランダムマスクを復元するため，計算コストは $\mathcal{O}((n-r)m + r(n-r)m)$ となる．

2.2 LightSecAgg

ランダムマスクを冗長化して交換する LightSecAgg は，ランダム鍵からのマスクの復元が不要なため計算コストが低いが，マスクの転送回数が多いため通信コストが高い．

各ユーザは自己ランダムマスクでローカルモデルをマスクしてからサーバへ送信する．サーバが生存ユーザのマスクされたローカルモデルを集約すると生存ユーザのランダムマスクが残るが，その和を次のように算出する．まず，各ユーザは t-private 最大距離分離符号 [6] を用いてランダムマスクから n 個の冗長ランダムマスクを生成する．次に，ユーザ間で冗長ランダムマスクを交換して集約する．最後に，サーバは生存ユーザの集約された冗長ランダムマスクを収集し，生存ユーザのランダムマスクの和を復元する．t-private 最大距離分離符号は，$t+1$ 未満のセミオネストなユーザが，任意のオネストなユーザの冗長ランダムマスクからそのユーザのランダムマスクを復元することを防ぐ．また，この符号は，離脱ユーザ数が r が $n-(t+1)$ 以下であればサーバがランダムマスクの和を復元できる．

サーバはランダムマスクの和のみ復元するため，計算コストは $\mathcal{O}(m)$ となる．しかし，各ユーザは冗長ランダムマスクを m 次元のベクトルとして $n-1$ 台のユーザに送信するため，通信コストは $\mathcal{O}(nm)$ となる．

2.3 その他のプロトコル

SecAgg の計算コストや LightSecAgg の通信コストを改善するプロトコルが提案されている [4], [7], [8], [9], [10], [11], [12] が，プライバシー保護やドロップアウト耐性の強度が低下する．本研究では，強度を損なわずに計算と通信コストを改善する．

3. 問題設定

3.1 システムモデル

記号の説明は**表 2** の通りである．n 台のユーザの集合 $\mathcal{U}$ とサーバが以下のラウンドを繰り返してグローバルモデルを学習する．各ラウンドでは，各ユーザ $i \in \mathcal{U}$ はサーバから次元 m のベクトルであるグローバルモデル $\mathbf{w}$ を受信し，各ユーザ i は自身の教師データを用いて $\mathbf{w}$ を学習し，ローカルモデル $\mathbf{w}_i$ を生成する．その後，ユーザとサーバはセキュアアグリゲーションプロトコルによってローカルモデルを共有せずに集約する．最後に，サーバはローカルモデルの和を用いて $\mathbf{w}$ を更新する．

本研究では，Bonawitz ら [3] と同じ通信モデルを仮定する．このモデルでは，1) ユーザ間の通信はサーバが仲介する，2) ネットワークの混雑やバッテリーの低下などにより，一定数（r）のユーザが任意のタイミングで離脱しうる．サーバへ（マスクされた）ローカルモデルを送信しない離脱ユーザの集合を $\mathcal{D}$ とする．

3.2 脅威モデル

t 台のユーザの集合 $\mathcal{T}$ とサーバは結託した攻撃者であり，任意のオネストなユーザ $h \in \mathcal{U} \setminus \mathcal{T}$ のローカルモデルを取得することを目的とする．攻撃者はセミオネストであり，プロトコルに従うが，プロトコルで受信した情報を共有し，共有情報を用いてローカルモデルの復元を試みる．

3.3 目標

プライバシー保護の目標は，プロトコルで受信した情報から，攻撃者が任意のオネストなユーザのローカルモデルを復元できないことである．ドロップアウト耐性の目標は，一部のユーザが離脱しても，サーバが生存ユーザのローカルモデルの和 $\sum_{i \in \mathcal{U} \setminus \mathcal{D}} \mathbf{w}_i$ を算出できることである．

4. BalancedSecAgg

4.1 プリミティブ

4.1.1 鍵合意

鍵合意 (Key Agreement) スキームは，アルゴリズムの組 $(\mathbf{KA.param}, \mathbf{KA.gen}, \mathbf{KA.agree})$ から構成される．セキュリティパラメータ κ に対し，$\mathbf{KA.param}$ は公開パラメータ pp を生成し $(\mathbf{KA.param}(\kappa) \to pp)$，$\mathbf{KA.gen}$ は各ユーザ i は公開鍵と秘密鍵のペア (pk_i, sk_i) を生成し $(\mathbf{KA.gen}(pp) \to (pk_i, sk_i))$, $\mathbf{KA.agree}$ は, 各ユーザ i の秘

密鍵 sk_i と他のユーザ j の公開鍵 pk_j から，ユーザ i とユーザ j の共有鍵 $a_{i,j}$ を生成する ($\mathbf{KA.agree}(sk_i, pk_j) \to a_{i,j}$)．

正当性 (Correctness) として，任意のユーザ i と j のペアに対し，$\mathbf{KA.agree}(sk_i, pk_j) = \mathbf{KA.agree}(sk_j, pk_i) = a_{i,j}$ が成り立つことを要求する．安全性として，i と j の公開鍵が与えられたセミオネストな攻撃者に対し，$a_{i,j}$ が一様ランダムな文字列と区別できないことを要求する．本研究では，Diffie-Hellman 鍵合意 [13] を使用するため，安全性は Decisional Diffie-Hellman (DDH) 仮定に依存する．

4.1.2 疑似乱数生成器

セキュアな疑似乱数生成器 (**PRG**) [14], [15] は，ある一様ランダムな鍵 s を入力として受け取り，$[0, p)$ の範囲内の数値からなる列 $\mathbf{PRG}(s)$ を生成する．安全性として，鍵が隠蔽されている限り，出力が同じ範囲から一様に選ばれた真の乱数列と計算量的に識別不能であることを要求する．

4.1.3 認証付き暗号

認証付き暗号 [16] は 2 者間で交換するメッセージの機密性と完全性を保証する．これは，鍵を用いてメッセージを暗号化する暗号化アルゴリズム **AE.enc** と，鍵を用いて暗号文を復号して元の平文を復元するか，エラー記号 ($\perp$) を出力する復号アルゴリズム **AE.dec** で構成される．

正当性として，すべての鍵 $k \in \{0,1\}^\kappa$ およびすべてのメッセージ x に対して，$\mathbf{AE.dec}(k, \mathbf{AE.enc}(k, x)) = x$ が成り立つことを要求する．安全性として，選択平文攻撃 (CPA) に対する識別不能性（IND-CPA）および暗号文の完全性（IND-CTXT）を要求する [16].

4.1.4 シャミアの秘密分散

$(t+1, n)$-しきい値シャミアの秘密分散 [5] によって秘密値 s から生成される n 個のシェアは，任意の $t+1$ 個からは s を復元でき，t 個以下からは s に関する情報が得られないという特徴をもつ．シャミアの秘密分散は，シェア生成アルゴリズム **SS.share**，秘密値復元アルゴリズム **SS.recon** から構成される．$\mathbf{SS.share}(s, t+1, \mathcal{U}) \to \{(i, sh_i^s)\}_{i\in\mathcal{U}}$ は，秘密 s，しきい値 $t+1$，n 個の要素からなる集合 $\mathcal{U}$ を入力とし，n 個のシェアを出力する．具体的には，$f(0) = s$ となる有限体 $\mathbb{F}$ 上のランダムな多項式 $f \in \mathbb{F}[X]$ を選び，評価点をシェアとする．$\mathbf{SS.recon}(\{(i, sh_i^s)\}_{i\in\mathcal{V}}, t+1) = s$ は，部分集合 $\mathcal{V} \subseteq \mathcal{U}$ に対応するシェアとしきい値 $t+1$ を入力とし，s を出力する．ただし，$|\mathcal{V}| \geq t+1$ である．

正当性として，$\forall s \in \mathbb{F}, \forall t+1, n$ に対し，$1 \leq t+1 \leq n$ かつ $|\mathcal{U}| = n$ である $\mathcal{U} \subseteq \mathbb{F}$ について，$\mathbf{SS.share}(s, t+1, \mathcal{U}) \to \{(i, sh_i^s)\}_{i\in\mathcal{U}}$ が成り立つとき，$\mathcal{V} \subseteq \mathcal{U}$ かつ $|\mathcal{V}| \geq t+1$ であれば，$\mathbf{SS.recon}(\{(i, sh_i^s)\}_{i\in\mathcal{V}}, t+1) = s$ が成り立つことを要求する．安全性として，$\forall s, s' \in \mathbb{F}$ および任意の $t+1$ 未満のシェアに対して，以下の条件を要求する：

$$\{\mathbf{SS.share}(s, t+1, \mathcal{U}) \to \{(i, sh_i^s)\}_{i\in\mathcal{U}} : \{(i, sh_i^s)\}_{i\in\mathcal{V}}\} \equiv \{\mathbf{SS.share}(s', t+1, \mathcal{U}) \to \{(i, sh_i^{s'})\}_{i\in\mathcal{U}} : \{(i, sh_i^{s'})\}_{i\in\mathcal{V}}\}$$

Algorithm 1 Select algorithm on a user i

```
1: procedure Select(i, n, t+1, U_1)
2:     S_i ← {}                                   ▷ Initialize
3:     c ← t+1
4:     for a = 1, 2, ··· , n−1 do
5:         b ← (i + a) mod n;
6:         if b ∈ U_1 then
7:             S_i ← S_i ∪ {b}
8:             c ← c − 1
9:             if c = 0 then
10:                break;
11:            end if
12:        end if
13:    end for
14:    return S_i
15: end procedure
```

図 1: ユーザ i による **Select** アルゴリズム
Fig. 1 **Select** algorithm on a user i

ここで，$\equiv$ は 2 つの分布が同一であることを示す．

重要な特徴が 2 つある．第 1 に，加算に関する線形性をもつ．具体的には，2 つの秘密値 s_a のシェアの集合 $\{(i, sh_i^{s_a})\}_{i\in\mathcal{U}}$ と s_b のシェアの集合 $\{(i, sh_i^{s_b})\}_{i\in\mathcal{U}}$ に対し，$\{(i, sh_i^{s_a} + sh_i^{s_b})\}_{i\in\mathcal{U}}$ を秘密値 $s_a + s_b$ のシェアの集合と見なせる．第 2 に，**SS.share** によって生成された n 個のシェアを，多項式に基づくリードソロモン消失訂正符号における符号語を構成する n 個のシンボルと見なせる [17].

4.1.5 リードソロモン消失訂正符号

リードソロモン消失訂正符号 [18] は符号化アルゴリズム **RS.encode** と復号アルゴリズム **RS.decode** で構成される．$\mathbf{RS.encode}(s_1, s_2, \ldots, s_{t+1}, n) \to c_1, c_2, \ldots, c_n$ は，$t+1$ 個のシンボル $(s_1, s_2, \ldots, s_{t+1})$ と符号語長 n を入力とし，符号語を構成する n 個のシンボル $(c_1, c_2, \ldots, c_n)$ を出力する．$\mathbf{RS.decode}(\{c_i\}_{i\in\mathcal{V}}, t+1, n) = s_1, s_2, \ldots, s_{t+1}$ は，$t+1$ 以上のシンボルを含む集合 $\mathcal{V}$ に対応する符号語のシンボル，$t+1$，n を入力とし，元のシンボルを出力する．

重要な特徴が 2 つある．第 1 に，**RS.decode** に含まれる消失訂正関数 **RS.correct** が，符号語のうち，消失した $n-(t+1)$ 個のシンボルを生成する点である．$(c_1, c_2, \ldots, c_n)$ に対し，$\mathbf{RS.correct}(\{c_i\}_{i\in\mathcal{V}}, t+1) \to \{c_j\}_{j\in[n]\setminus\mathcal{V}}$ は，$t+1$ 以上のシンボル c_i を含む任意の集合 $\mathcal{V}$ を入力とし，残りの $n-(t+1)$ 個のシンボルの集合を出力する．この関数は，$t+1$ 個のシンボルから追加の $n-(t+1)$ 個のシンボルを生成する関数と見なせる．本研究では，$t+1$ 個のシンボルを入力シンボル，追加の $n-(t+1)$ 個のシンボルを冗長シンボルと呼ぶ．第 2 に，シャミアの秘密分散と同様に加算に関する線形性を持つ．2 つのシンボル集合 $\{c_i^a\}_{i\in\mathcal{V}}$ と $\{c_i^b\}_{i\in\mathcal{V}}$ に対し，これらの 2 つの集合の消失訂正結果を加算したものは，$\{c_i^a + c_i^b\}_{i\in\mathcal{V}}$ の消失訂正結果と同じになる．

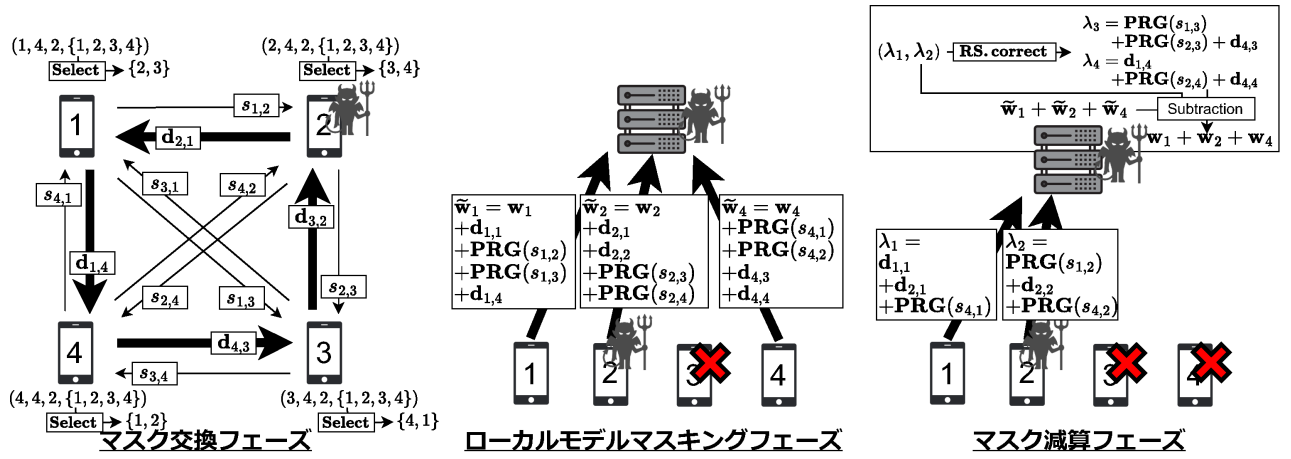

図 2: BalancedSecAgg の例．$n = 4$, ユーザ 2 はセミオネストユーザ, ユーザ 3, 4 は離脱ユーザである ($t = 1, r = 2, \mathcal{D} = \{3\}$).

Fig. 2 An example of BalancedSecAgg with $n = 4$ users, where user 2 is a semi-honest user ($t = 1$) and user 3 and user 4 are dropout users ($r = 2, \mathcal{D} = \{3\}$).

4.2 設計根拠

主なアイデアは n 個のマスクに対応する要素を，リードソロモン消失訂正符号における符号語を構成する n 個のシンボルとして捉えることである．各 n 個のシンボルは，$t+1$ 個の入力シンボルと $n-(t+1)$ 個の冗長シンボルで構成される．入力シンボルを含むマスクをランダムマスク，冗長シンボルを含むマスクを冗長ランダムマスクと呼ぶ．各ユーザは $t+1$ 個のランダムマスクと $n-(t+1)$ 個の冗長ランダムマスクを自身のローカルモデルに加えてからサーバに送信する．その後，ユーザ間でランダムマスクと冗長ランダムマスクを交換し，集約する．サーバは生存ユーザの集約されたマスクを収集し，それらを用いて生存ユーザのローカルモデルの和を計算する．

4.3 プロトコル

この章ではプロトコルの要点を説明する．プロトコルの詳細は図 3 に示している．

4.3.1 要点

各ユーザ i は, $t+1$ 個のランダムマスク $\{\mathbf{PRG}(s_{i,j})\}_{j\in\mathcal{S}_i}$ ($|\mathcal{S}_i| = t+1$) を生成し，さらに **RS.correct** によって $n-(t+1)$ 個の冗長ランダムマスク $\{\mathbf{d}_{i,k}\}_{k\in\mathcal{U}\setminus\mathcal{S}_i}$ ($|\mathcal{U}\setminus\mathcal{S}_i| = n-(t+1)$) を以下のように生成する：

$$\{\mathbf{d}_{i,k}\}_{k\in\mathcal{U}\setminus\mathcal{S}_i} = \mathbf{RS.correct}(\{\mathbf{PRG}(s_{i,j})\}_{j\in\mathcal{S}_i}, t+1).$$

ここで，$\mathcal{S}_i$ は図 1 に示す **Select** によって生成されたユーザの部分集合であり，ユーザ i からランダム鍵 $s_{i,j}$ を受信するユーザの集合である．**Select** はユーザ間の冗長ランダムマスクの交換を均等に負荷分散するアルゴリズムである．

次に，各ユーザ i はマスクされたローカルモデル $\widetilde{\mathbf{w}}_i$ を以下のように生成する：

$$\widetilde{\mathbf{w}}_i = \mathbf{w}_i + \sum_{j\in\mathcal{S}_i}\mathbf{PRG}(s_{i,j}) + \sum_{k\in\mathcal{U}\setminus\mathcal{S}_i}\mathbf{d}_{i,k} \pmod p. \quad (1)$$

最後に，各ユーザ i は $\widetilde{\mathbf{w}}_i$ をサーバへ送信する．

サーバは生存ユーザのマスクされたローカルモデルを収集し，その和から生存ユーザのマスクの和を減算することで生存ユーザのローカルモデルの和を算出する．

これを達成するために，各ユーザ i は各ユーザ $j \in (\mathcal{U}\setminus\mathcal{D})\cap\mathcal{S}_i$ にランダム鍵 $s_{i,j}$ を，各ユーザ $k \in (\mathcal{U}\setminus\mathcal{D})\setminus\mathcal{S}_i$ に冗長ランダムマスク $\mathbf{d}_{i,k}$ を送信する．その後，各ユーザ i は生存ユーザのマスクを以下のように集約する：

$$\lambda_i = \sum_{j\in(\mathcal{U}\setminus\mathcal{D})\cap\{k|i\in\mathcal{S}_k\}}\mathbf{PRG}(s_{j,i}) + \sum_{j\in(\mathcal{U}\setminus\mathcal{D})\setminus\{k|i\in\mathcal{S}_k\}}\mathbf{d}_{j,i} \pmod p, \quad (2)$$

最後に，各ユーザ i は λ_i をサーバへ送信する．

サーバはは収集した $\{\lambda_j\}_{j\in\mathcal{U}\setminus\mathcal{D}}$ から **RS.correct** によって離脱ユーザが生成する生存ユーザの集約されたマスクを復元できる（$\{\lambda_k\}_{k\in\mathcal{D}} = \mathbf{RS.correct}(\{\lambda_j\}_{j\in\mathcal{U}\setminus\mathcal{D}}, t+1)$. これは $t+1$ 個の集約されたマスクは $\{\lambda_j\}_{j\in\mathcal{U}}$ に含まれる $t+1$ 個のランダムマスクと見なせるためである．最後に，サーバは生存ユーザのローカルモデルの和 $\mathbf{v} = \sum_{i\in\mathcal{U}\setminus\mathcal{D}}\mathbf{w}_i$ を以下のように計算する：

$$\begin{aligned}\mathbf{v} &= \sum_{i\in\mathcal{U}\setminus\mathcal{D}}\widetilde{\mathbf{w}}_i - \sum_{i\in\mathcal{U}}\lambda_i \\ &= \sum_{i\in\mathcal{U}\setminus\mathcal{D}}(\mathbf{w}_i + \sum_{j\in\mathcal{S}_i}\mathbf{PRG}(s_{i,j}) + \sum_{j\in\mathcal{U}\setminus\mathcal{S}_i}\mathbf{d}_{i,j}) \\ &\quad - \sum_{i\in\mathcal{U}}(\sum_{j\in(\mathcal{U}\setminus\mathcal{D})\cap\{k|i\in\mathcal{S}_k\}}\mathbf{PRG}(s_{j,i}) + \sum_{j\in(\mathcal{U}\setminus\mathcal{D})\setminus\{k|i\in\mathcal{S}_k\}}\mathbf{d}_{j,i}) \pmod p. \quad (3)\end{aligned}$$

4.3.2 例

図 2 に示す例を用いてプロトコルを説明する．この例では，$n = 4$，ユーザ 2 がセミオネストなユーザ（$t = 1$），ユーザ 3 と 4 が離脱ユーザ（$r = 2, \mathcal{D} = \{3\}$）である．図 2 の細い線と太い線は，それぞれスカラーとベクトルを示す．

- **セットアップフェーズ**
  - 全参加者には，セキュリティパラメータ κ，全ユーザの数 n，セミオネストユーザの数 t，ドロップアウトユーザの数 r，鍵合意パラメータ $pp \leftarrow \mathbf{KA.param}(\kappa)$，および有限体 $\mathbb{F}_p$ が与えられる．各ユーザはサーバとの間にセキュアチャネルを確立している．

  ユーザ i:
  - 公開鍵秘密鍵ペア (pk_i, sk_i) を生成 $((pk_i, sk_i) \leftarrow \mathbf{KA.gen}(pp))$.
  - サーバへ公開鍵 pk_i を送信.

  サーバ:
  - $t+2$ 台以上のユーザの公開鍵を収集する．そうでなければプロトコルを中止する．公開鍵を送信したユーザの集合を $\mathcal{U}_1 (\subseteq \mathcal{U})$ とする．
  - $\mathcal{U}_1$ に含まれる全ユーザへ公開鍵のリスト $\{j, pk_j\}_{j\in\mathcal{U}_1}$ を送信する．
- **マスク交換フェーズ:**

  ユーザ i:
  - サーバからリスト $\{j, pk_j\}_{j\in\mathcal{U}_1}$ を受信する．$|\mathcal{U}_1| \geq t+2$ であることを確認し，それに満たない場合，プロトコルを中止する．
  - **Select** によって，ユーザ i のランダム鍵を受信するユーザの集合 $\mathcal{S}_i$ を生成する $(\mathcal{S}_i \leftarrow \mathbf{Select}(i, n, t+1, \mathcal{U}_1))$.
  - 各ユーザ $j \in \mathcal{S}_i$ に対し，ランダム鍵 $s_{i,j}$ を生成し，$\{s_{i,j}\}_{j\in\mathcal{S}_i}$ から **PRG** を用いてランダムマスク $\{\mathbf{PRG}(s_{i,j})\}_{j\in\mathcal{S}_i}$ を生成する．
  - $\mathcal{U}_1 \setminus \mathcal{S}_i$ に属するユーザに対して，冗長ランダムマスク $\{\mathbf{d}_{i,k}\}_{k\in\mathcal{U}_1\setminus\mathcal{S}_i} = \mathbf{RS.correct}(\{\mathbf{PRG}(s_{i,j})\}_{j\in\mathcal{S}_i}, t+1)$ を生成する．
  - 各ユーザ $j \in \mathcal{S}_i$ に対して，暗号文 $c_{i,j} \leftarrow \mathbf{AE.enc}(\mathbf{KA.agree}(sk_i, pk_j), i||j||s_{i,j})$ を生成する．
  - 各ユーザ $k \in \mathcal{U}_1 \setminus \mathcal{S}_i$ に対して，暗号文 $c_{i,k} \leftarrow \mathbf{AE.enc}(\mathbf{KA.agree}(sk_i, pk_k), i||k||\mathbf{d}_{i,k})$ を生成する．
  - 全暗号文 $\{c_{i,j}\}_{j\in\mathcal{U}_1\setminus\{i\}}$ をサーバへ送信する．

  サーバ:
  - $t+2$ 台以上のユーザの暗号文を収集する．そうでなければ，プロトコルを中止する．暗号文を送信したユーザの集合を $\mathcal{U}_2 (\subseteq \mathcal{U}_1)$ とする．
  - 各ユーザ $i \in \mathcal{U}_2$ に対して，ユーザ i 宛の全ての暗号文のリスト $\{c_{j,i}\}_{j\in\mathcal{U}_2}$ を送信する．
- **ローカルモデルマスキングフェーズ:**

  ユーザ i:
  - サーバからリスト $\{c_{j,i}\}_{j\in\mathcal{U}_2}$ を受信する．もし $|\mathcal{U}_2|$ が $t+2$ 未満の場合, プロトコルを中止する.
  - マスクされたローカルモデル $\widetilde{\mathbf{w}}_i = \mathbf{w}_i + \sum_{j\in\mathcal{S}_i} \mathbf{PRG}(s_{i,j}) + \sum_{k\in\mathcal{U}_1\setminus\mathcal{S}_i} \mathbf{d}_{i,k}$ を生成し，サーバへ送信する．

  サーバ:
  - $t+2$ 台以上のユーザからマスクされたローカルモデルを収集する．そうでなければプロトコルを中止する．マスクされたローカルモデルを送信したユーザの集合を $\mathcal{U}_3 (\subseteq \mathcal{U}_2)$ とする．
  - 各ユーザ $i \in \mathcal{U}_3$ に対して，$\mathcal{U}_3$ のリストを送信する．
- **マスク減算フェーズ:**

  ユーザ i:
  - サーバから $\mathcal{U}_3$ を受信する．もし，$|\mathcal{U}_3|$ が $t+2$ 未満の場合，プロトコルを中止する．
  - 各ユーザ $j \in \mathcal{U}_3 \cap \{k \mid i \in \mathcal{S}_k\}$ に対して，暗号文 $c_{j,i}$ を復号する．$(j||i||s_{j,i} \leftarrow \mathbf{AE.dec}(\mathbf{KA.agree}(sk_i, pk_j), c_{j,i}))$.
  - 各ユーザ $k \in (\mathcal{U}_3 \setminus \{j \mid i \in \mathcal{S}_j\}) \setminus \{i\}$ に対して，暗号文 $c_{j,i}$ を復号する ($, k||i||\mathbf{d}_{k,i} \leftarrow \mathbf{AE.dec}(\mathbf{KA.agree}(sk_i, pk_k), c_{k,i})$).
  - ランダム鍵 $\{s_{j,i}\}_{j\in\mathcal{U}_3\cap\{k|i\in\mathcal{S}_k\}}$ から **PRG** を用いてランダムマスク $\{\mathbf{PRG}(s_{j,i})\}_{j\in\mathcal{U}_3\cap\{k|i\in\mathcal{S}_k\}}$ を生成する．
  - 集約されたマスク $\lambda_i = \sum_{j\in\mathcal{U}_3\cap\{k|i\in\mathcal{S}_k\}} \mathbf{PRG}(s_{j,i}) + \sum_{j\in\mathcal{U}_3\setminus\{k|i\in\mathcal{S}_k\}} \mathbf{d}_{j,i}$ を生成し，サーバへ送信する．

  サーバ:
  - $t+1$ 台以上のユーザの集約されたマスクを収集する．そうでなければプロトコルを中止する．集約されたマスクを送信したユーザの集合を $\mathcal{U}_4 (\subseteq \mathcal{U}_3)$ とする．
  - $\mathcal{U}_1 \setminus \mathcal{U}_4$ に含まれるユーザの集約されたマスクを復元する $(\{\lambda_k\}_{k\in\mathcal{U}_1\setminus\mathcal{U}_4} = \mathbf{RS.correct}(\{\lambda_j\}_{j\in\mathcal{U}_4}, t+1))$.
  - ローカルモデルの和 $\mathbf{v} = \sum_{i\in\mathcal{U}_3} \mathbf{w}_i = \sum_{i\in\mathcal{U}_3} \widetilde{\mathbf{w}}_i - \sum_{i\in\mathcal{U}_1} \lambda_i$ を算出する．

図 3: BalancedSecAgg の詳細な説明

Fig. 3 Detailed description of BalancedSecAgg.

マスク交換フェーズ． このフェーズでは，各ユーザ $i \in \{1,2,3,4\}$ は，**Select** によりランダム鍵を送信するユーザの集合 $\mathcal{S}_i$ を生成する．次に，ユーザ i はランダムマスク $\{\mathbf{PRG}(s_{i,j})\}_{j\in\mathcal{S}_i}$ と冗長ランダムマスク $\{\mathbf{d}_{i,k}\}_{k\in\mathcal{U}\setminus\mathcal{S}_i}$ を生成し，$\mathcal{S}_i$ に属する各ユーザ j にランダム鍵 $s_{i,j}$ を，それ以外の各ユーザ k に冗長ランダムマスク $\mathbf{d}_{i,k}$ を送信する．

例えば，図 2 のユーザ 1 は $i=1, n=4, t+1=2$, $\mathcal{U}_1 = \{1,2,3,4\}$ を **Select** に入力し，$\mathcal{S}_1 = \{2,3\}$ を得る．次に，ユーザ 1 は $\mathbf{PRG}(s_{1,2})$, $\mathbf{PRG}(s_{1,3})$, $\mathbf{d}_{1,4}$ を生成し，ユーザ 2, 3 に $s_{1,2}$, $s_{1,3}$ を，ユーザ 4 に $\mathbf{d}_{1,4}$ を送信する．

ローカルモデルマスキングフェーズ． このフェーズでは，各ユーザ $i \in \{1,2,4\}$ が式 (1) に従って生成したマスクされたローカルモデルをサーバに送信する．

例えば, 図 2 のユーザ 1 は $\widetilde{\mathbf{w}}_1 = \mathbf{w}_1 + \mathbf{d}_{1,1} + \mathbf{PRG}(s_{1,2}) + \mathbf{PRG}(s_{1,3}) + \mathbf{d}_{1,4}$ をサーバへ送信する．

マスク減算フェーズ． このフェーズでは, 各ユーザ $i \in \{1,2\}$ は式 (2) に従って集約したマスクをサーバへ送信し，サーバはローカルモデルの和を式 (3) に従って計算する．

例えば，図 2 のユーザ 1 は，ユーザ 4 から受信した $s_{4,1}$ から $\mathbf{PRG}(s_{4,1})$ を生成し，$\lambda_1 = \mathbf{d}_{1,1} + \mathbf{d}_{2,1} + \mathbf{PRG}(s_{4,1})$ を生成してサーバへ送信する．サーバは受信する λ_1 と λ_2 から，**RS.correct** によって λ_3 と λ_4 を復元し，$\mathbf{v} = \sum_{i\in\{1,2,4\}} \widetilde{\mathbf{w}}_i - \sum_{i\in\{1,2,3,4\}} \lambda_i$ を計算することでローカルモデルの和 $\mathbf{v} = \sum_{i\in\{1,2,4\}} \mathbf{w}_i$ を取得する．

5. 安全性証明

この章では，BalancedSecAgg が 3 章で述べた目標を達成することを証明する．サーバと最大 t 台のセミオネストなユーザが共有する情報（view と呼ぶ）が，プロトコルの出力（ローカルモデルの和）から推測できる情報を超えて，オネストなユーザの入力（ローカルモデル）に関する情報を漏らさないことを証明する．view は，入力，ランダム性（randomness），公開情報，他のプロトコル参加者から受信したすべてのメッセージで構成される．参加者がプロトコルを中断した場合，メッセージの受信が停止する．

いくつかの記号を導入する．サーバを S とする．ユーザの部分集合 $\mathcal{U}' \subseteq \mathcal{U}$ のローカルモデルの集合を $\mathbf{w}_{\mathcal{U}'} = \{\mathbf{w}_i\}_{i\in\mathcal{U}'}$ とする．ユーザの部分集合 $\mathcal{T} \subset \mathcal{U}$ が与えられたとき，$\mathcal{T} \cup S$ に属するすべての参加者の view を表す確率変数を $\mathrm{REAL}_{\mathcal{T}}^{\mathcal{U},t,\kappa}(\mathbf{w}_{\mathcal{U}},\mathcal{U}_1,\mathcal{U}_2,\mathcal{U}_3,\mathcal{U}_4)$ とする．

次の定理では，$\mathcal{T} \cup S$ に属するすべての参加者の view $\mathrm{REAL}_{\mathcal{T}}^{\mathcal{U},t,\kappa}$ は，セミオネストなユーザの入力と，オネストなユーザのローカルモデルの合計でシミュレートできることを証明する．これは，攻撃者がシミュレートに用いる情報以外は知ることができないことを意味する．さらに，多くのユーザがローカルモデルマスキングフェーズを完了する前に中止した場合，オネストなユーザの入力を用いることなく $\mathrm{REAL}_{\mathcal{T}}^{\mathcal{U},t,\kappa}$ をシミュレートできることを証明する．

定理 1. 確率的多項式時間シミュレータ SIM が存在し，任意の t, $\mathbf{w}_{\mathcal{U}}$, $\mathcal{U}_1$, $\mathcal{U}_2$, $\mathcal{U}_3$, $\mathcal{U}_4$, $\mathcal{T}$ に対し，$\mathcal{T} \subset \mathcal{U}$, $|\mathcal{T}\setminus S| \le t$, $\mathcal{U} \supseteq \mathcal{U}_1 \supseteq \mathcal{U}_2 \supseteq \mathcal{U}_3 \supseteq \mathcal{U}_4$ が成り立つとき，SIM の出力は $\mathrm{REAL}_{\mathcal{T}}^{\mathcal{U},t,\kappa}$ と計算量的に識別不能である．すなわち，

$$\begin{aligned}&\mathrm{REAL}_{\mathcal{T}}^{\mathcal{U},t,\kappa}(\mathbf{w}_{\mathcal{U}},\mathcal{U}_1,\mathcal{U}_2,\mathcal{U}_3,\mathcal{U}_4)\\ \approx&\mathrm{SIM}_{\mathcal{T}}^{\mathcal{U},t,\kappa}(\mathbf{w}_{\mathcal{T}},\mathbf{v},\mathcal{U}_1,\mathcal{U}_2,\mathcal{U}_3,\mathcal{U}_4),\end{aligned}$$

ここで

$$\mathbf{v} = \begin{cases}\sum_{i\in\mathcal{U}_3\setminus\mathcal{T}} \mathbf{w}_i & \textit{if } |\mathcal{U}_3| \ge t+2,\\ \perp & \textit{otherwise.}\end{cases}$$

証明． ハイブリッド論法で証明する．SIM を REAL から段階的に変更し，それらの確率変数が計算量的に識別不能であることを示す．

Hybrid$_0$: REAL と同じである．

Hybrid$_1$: オネストなすべてのユーザ $i \in \mathcal{U}_2 \setminus \mathcal{T}$ に対し，$j \in \mathcal{U}_1 \setminus \mathcal{T}$ に属する他のオネストなユーザ j との共有鍵 $\mathbf{KA}.\mathbf{agree}(sk_i, pk_j)$ を SIM が選択した一様ランダムな鍵に置き換える．DDH 仮定により，このハイブリッドは **Hybrid$_0$** と識別不能である．

Hybrid$_2$: オネストなすべてのユーザ $i \in \mathcal{U}_2 \setminus \mathcal{T}$ に対し，ユーザ i が他のオネストなユーザ $j \in \mathcal{U}_1 \setminus \mathcal{T}$ に送信したすべてのランダム鍵と冗長ランダムマスクの暗号文を，ランダム値やランダムマスク（例えば，0 やゼロベクトル）の暗号文に置き換える．ただし，その集合に属するオネストなユーザは，マスク減算フェーズでは引き続き正しいランダム鍵と冗長ランダムマスクを使用する．IND-CPA により，このハイブリッドは **Hybrid$_1$** と識別不能である．

Hybrid$_3$: $\mathcal{U}^*$ を次のように定義する：

$$\mathcal{U}^* = \begin{cases}\mathcal{U}_2 \setminus \mathcal{T} & \text{if } \mathbf{v} = \perp,\\ \mathcal{U}_2 \setminus \mathcal{U}_3 \setminus \mathcal{T} & \text{otherwise.}\end{cases}$$

すべてのオネストなユーザ $i \in \mathcal{U}^*$ に対し，他のオネストなユーザ $j \in \mathcal{S}i \setminus \mathcal{T}$ と共有するランダムマスク $\mathbf{PRG}(s_{i,j})$ を一様ランダムなマスク $\mathbf{u}_{i,j}$ に置き換える．**PRG** の安全性により，このハイブリッドは **Hybrid$_2$** と識別不能である．

Hybrid$_4$: すべてのオネストなユーザ $i \in \mathcal{U}$ に対し，

$$\widetilde{\mathbf{w}}_i = \mathbf{w}_i + \sum_{j\in\mathcal{S}_i\setminus\mathcal{T}} \mathbf{u}_{i,j} + \sum_{j\in\mathcal{S}_i\cap\mathcal{T}} \mathbf{PRG}(s_{i,j}) + \sum_{j\in\mathcal{U}_1\setminus\mathcal{S}_i} \mathbf{d}_{i,j}$$

を

$$\widetilde{\mathbf{w}}_i = \sum_{j\in\mathcal{S}_i\setminus\mathcal{T}} \mathbf{u}_{i,j} + \sum_{j\in\mathcal{S}_i\cap\mathcal{T}} \mathbf{PRG}(s_{i,j}) + \sum_{j\in\mathcal{U}_1\setminus\mathcal{S}_i} \mathbf{d}_{i,j}$$

に置き換える．$\mathbf{w}_i + \sum_{j\in\mathcal{S}_i\setminus\mathcal{T}} \mathbf{u}_{i,j}$ が一様ランダムであるため，このハイブリッドは **Hybrid$_3$** と識別不能である．さらに，このハイブリッドおよびそれ以降のすべてのハイブリッドは，$\mathcal{U}^*$ に属するユーザ i の $\mathbf{w}_i$ に依存しない．

もし $\mathbf{v}$ が $\perp$ であれば，SIM は **Hybrid$_4$** であり，SIM は $\mathcal{U} \setminus \mathcal{T}$ に属するすべてのオネストなユーザの入力を用いずに REAL をシミュレートできる．したがって，以下のハイブリッドでは $\mathbf{v} \neq \perp$ であると仮定する．

Hybrid$_5$: すべてのオネストなユーザ $i \in \mathcal{U}_2 \setminus \mathcal{T}$ に対し，

$$\{\mathbf{d}_{i,k}\}_{k\in\mathcal{U}_1\setminus\mathcal{S}_i} = \mathbf{RS}.\mathbf{correct}(\{\mathbf{PRG}(s_{i,j})\}_{j\in\mathcal{S}_i}, t+1)$$

を

$$\begin{aligned}\{\mathbf{q}_{i,k}\}_{k\in\mathcal{U}_1\setminus\mathcal{S}_i} = \mathbf{RS}.\mathbf{correct}(&\{\mathbf{u}_{i,j}\}_{j\in\mathcal{S}_i\setminus\mathcal{T}}\\ &\cup\{\mathbf{PRG}(s_{i,j})\}_{j\in\mathcal{S}_i\cap\mathcal{T}}, t+1),\end{aligned}$$

に置き換える．ここで，各 $\mathbf{u}_{i,j}$ は一様ランダムなマスクである．2 つの集合 $\mathcal{M}_i^1 = \{\mathbf{d}_{i,k}\}_{k\in\mathcal{U}_1\setminus\mathcal{S}_i} \cup \{\mathbf{PRG}(s_{i,j})\}_{j\in\mathcal{S}_i\setminus\mathcal{T}} \cup \{\mathbf{PRG}(s_{i,j})\}_{j\in\mathcal{S}_i\cap\mathcal{T}}$ と $\mathcal{M}_i^2 = \{\mathbf{q}_{i,k}\}_{k\in\mathcal{U}_1\setminus\mathcal{S}_i} \cup \{\mathbf{u}_{i,j}\}_{j\in\mathcal{S}_i\setminus\mathcal{T}} \cup \{\mathbf{PRG}(s_{i,j})\}_{j\in\mathcal{S}_i\cap\mathcal{T}}$ は，それぞれ（一様ランダムな）秘密 y と y' の $|\mathcal{U}_1|$ 個のシェアの集合，すなわち $\mathbf{SS}.\mathbf{Share}(y, t+1, |\mathcal{U}_1|)$ と $\mathbf{SS}.\mathbf{Share}(y', t+1, |\mathcal{U}_1|)$ の出力と見なせる．セミオネストなユーザ全体の view には t 個のマスクしか含まれないため，$\mathcal{M}_i^1$ の任意の t 個のマスクの分布は，$\mathcal{M}_i^2$ の同数のマスクの分布と同一である．したがって，このハイブリッドは **Hybrid$_4$** と識別不能である．

Hybrid$_6$: 特定オネストなユーザ $\ell \in \mathcal{U}_3 \setminus \mathcal{T}$ を固定し，すべてのオネストなユーザが，**Select** で生成するユーザの集合に ℓ を含める．具体的には，ℓ にランダムマスクを送信しないオネストなユーザ i，すなわち $i \in (\mathcal{U}_1 \setminus \mathcal{T}) \cap \{k \mid \ell \notin \mathcal{S}_k\}$ に対し，$\mathcal{S}_i$ を次の式で表される $\mathcal{S}'_i$ に置き換える：

$$\mathcal{S}'_i = \mathcal{S}_i \setminus \{i^{\mathrm{sel}}\} \cup \{\ell\}.$$

ここで，i^{sel} は $\mathcal{S}_i$ から選んだ 1 台のユーザである．この変更により，$\mathcal{U}_2 \setminus \mathcal{T}$ に属するすべてのオネストなユーザが，ℓ に対してランダムマスクを送信できるようになる．ここで，$\mathcal{M}_i^3$ をユーザ i のマスクの集合 $\mathcal{M}_i^3 = \{\mathbf{q}_{i,k}\}_{k \in \mathcal{U}_1 \setminus \mathcal{S}'_i} \cup \{\mathbf{u}_{i,j}\}_{j \in \mathcal{S}'_i \setminus \mathcal{T}} \cup \{\mathbf{PRG}(s_{i,j})\}_{j \in \mathcal{S}'_i \cap \mathcal{T}}$ とする．セミオネストなユーザ全体の view には t 個のマスクしか含まれないため，$\mathcal{M}_i^2$ の任意の t 個のマスクの分布は，$\mathcal{M}_i^3$ の同数のマスクの分布と同一である．したがって，このハイブリッドは **Hybrid$_5$** と識別不能である．

Hybrid$_7$: すべてのオネストなユーザ $i \in \mathcal{U}_3 \setminus \mathcal{T}$ に対し，

$$\begin{aligned}\widetilde{\mathbf{w}}_i &= \mathbf{w}_i + \sum_{j \in \mathcal{S}'_i} \mathbf{u}_{i,j} + \sum_{j \in \mathcal{U}_1 \setminus \mathcal{S}'_i} \mathbf{d}_{i,j} \\ &= \mathbf{w}_i + \mathbf{u}_{i,\ell} + \sum_{j \in \mathcal{S}'_i \setminus \{\ell\}} \mathbf{u}_{i,j} + \sum_{j \in \mathcal{U}_1 \setminus \mathcal{S}'_i} \mathbf{d}_{i,j}\end{aligned}$$

を

$$\widetilde{\mathbf{w}}'_i = \mathbf{x}_i + \sum_{j \in \mathcal{S}'_i \setminus \{\ell\}} \mathbf{u}_{i,j} + \sum_{j \in \mathcal{U}_1 \setminus \mathcal{S}'_i} \mathbf{d}_{i,j},$$

に置き換える．ここで，$\{\mathbf{x}_i\}_{i \in \mathcal{U}_3 \setminus \mathcal{T}}$ は一様ランダムであり，$\sum_{i \in \mathcal{U}_3 \setminus \mathcal{T}} \mathbf{x}_i = \sum_{i \in \mathcal{U}_3 \setminus \mathcal{T}} (\mathbf{w}_i + \mathbf{u}_{i,\ell})$ を満たす．したがって，このハイブリッドは **Hybrid$_6$** と同一に分布しているため，**Hybrid$_6$** と識別不能である．SIM を **Hybrid$_7$** とすると，SIM は $\mathcal{U} \setminus \mathcal{T}$ に属するすべてのオネストなユーザの入力を用いずに REAL をシミュレートできる．以上の議論により，SIM の出力が REAL の出力と計算量的に識別不能であることが証明される． □

6. おわりに

リードソロモン消失訂正符号を用いてランダムマスクを冗長化することで SecAgg の計算コストと LightSecAgg の通信コストのトレードオフを調整するプロトコルを設計した．また，設計したプロトコルが SecAgg と LightSecAgg と同じ安全性の強度をもつことを証明した．

謝辞 本研究は，科研費 24K22295 によるものである．

参考文献

[1] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, ‘‘Federated learning in mobile edge networks: A comprehensive survey,’’ IEEE Commun. Surveys Tuts., vol. 22, no. 3, 3rd Quart., 2020, pp. 2031–2063.

[2] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, “Beyond inferring class representatives: User-level privacy leakage from federated learning,” in Proc. IEEE Conf. Comput. Commun. (INFOCOM), Apr. 2019, pp. 2512–2520.

[3] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. Mcmahan, S. Patel, D. Ramage, A. Segal, and K. Seth, “Practical secure aggregation for privacy-preserving machine learning,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur. New York, NY, USA: Association for Computing Machinery, Oct. 2017, pp. 1175–1191.

[4] J. So, C. He, C.-S. Yang, S. Li, Q. Yu, R. E. Ali, B. Guler, and S. Avestimehr, “ LightSecAgg: a lightweight and versatile design for secure aggregation in federated learning,”in Proc. Mach. Learn. Syst., vol. 4, Apr. 2022, pp. 694-720.

[5] A. Shamir, “How to share a secret,” Commun. ACM, vol. 22, no. 11, 1979, pp. 612–613.

[6] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and A. S. Avestimehr, ‘‘Lagrange coded computing: Optimal design for resiliency, security, and privacy,’’ in Proc. 22nd Int. Conf. Artif. Intell. Statist. (AISTATS), Okinawa, Japan, Apr. 2019, pp. 1215–1225.

[7] J. H. Bell, K. Bonawitz, A. Gascón, T. Lepoint, and M. Raykova, “Secure single-server aggregation with (poly) logarithmic overhead,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., Oct. 2020, pp. 1253–1269.

[8] B. Choi, J.-y. Sohn, D.-J. Han, and J. Moon, “Communication-Computation Efficient Secure Aggregation for Federated Learning,”2020, arXiv:2012.05433.

[9] E. van Kempen, Q. Li, G. A. MArson, and C. Soriente, “Lisa: Lightweight single-server secure aggregation with a public source of randomness,”2023, arXiv:2308.02208.

[10] J. So, B. Guler, and A. S. Avestimehr, “Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning,” IEEE J. Sel. Areas Inf. Theory, vol. 2, no. 1, Mar. 2021, pp. 479–489.

[11] T. Jahani-Nezhad, M. A. Maddah-Ali, S. Li, and G. Caire, “Swiftagg+: Achieving asymptotically optimal communication load in secure aggregation for federated learning,” J. Sel. Areas Commun., vol. 41, no. 4, 2023, pp.977-989.

[12] S. Kadhe, N. Rajaraman, O. O. Koyluoglu, and K. Ramchandran, “FastSecAgg: Scalable secure aggregation for privacy-preserving federated learning,” 2020, arXiv:2009.11248.

[13] W. Diffie and M. Hellman, ‘‘New directions in cryptography,’’ IEEE Trans. Inf. Theory, vol. IT-22, no. 6, Nov. 1976, pp. 644–654.

[14] M. Blum and S. Micali, ‘‘How to generate cryptographically strong sequences of pseudo-random bits,’’ SIAM J. Comput., vol. 13, no. 4, 1984, pp. 850–864.

[15] A. C. Yao, ‘‘Theory and application of trapdoor functions,’’ in Proc. 23rd Annu. Symp. Found. Comput. Sci. (sfcs), Nov. 1982, pp. 80–91.

[16] M. Bellare and C. Namprempre, ‘‘Authenticated encryption: Relations among notions and analysis of the generic composition paradigm,’’ in Advances in Cryptology—ASIACRYPT. Berlin, Germany: Springer, 2000, pp. 531–545.

[17] R. J. McEliece and D. V. Sarwate, ‘‘On sharing secrets and reed-Solomon codes,’’ Commun. ACM, vol. 24, no. 9, Sep. 1981, pp. 583–584.

[18] S. Lin and D. J. Costello, ‘‘Error control coding: fundamentals and applications,’’ Pearson/Prentice Hall, Upper Saddle River, NJ, 2004.