Vol.2022-AL-187 No.7
2022/3/14

秘密計算基数ソートの通信量の削減

吉田 勇輝[1,a)]　定兼 邦彦[2,b)]

概要：対象のデータの情報を秘匿したまま計算を行う秘密計算において，特にソーティングは多くのデータ処理において必要かつボトルネックであり，その高速化は重要な課題である．本稿では，3 パーティの秘密分散による semi-honest 安全な秘密計算基数ソートについて，サブプロトコルの計数ソートを既存手法の組み合わせにより効率化し，オンライン通信量の削減による高速化を行う．また，semi-honest モデルでの安全性の証明についての考察を行う．今回仮定するモデル下での安全性の十分条件を導入し，出力がシェアであるプロトコルの安全性の証明を簡潔化する方針を説明する．

キーワード：秘密計算，秘密分散，ソーティング，基数ソート

Oblivious Radix Sort with Less Communication

Abstract: In secure computation, sorting protocol is necessary for many kinds of data analyses, and its acceleration is an important task. In this draft, by introducing an efficient counting sort protocol, we propose an oblivious radix sort with less online communication. This protocol is computed by 3-party and secure against semi-honest adversary. Moreover, we consider a method to prove security of some kinds of protocols in multi-party computation. We introduce sufficient conditions for security in some specific model and simplify the proof of protocols which output shares.

Keywords: secure computation, secret sharing, sorting, radix sort

1. はじめに

秘密計算は，対象のデータや計算結果についての情報を得ないまま計算を行う技術である．応用としては，個人情報などプライバシーに関わる情報や，企業や国家が持つ機密情報を共有しながら解析することなどが考えられる．その中でも，データ解析においてソーティングは多くの処理で必要かつボトルネックな操作である．例えば，様々な統計量の導出やデータの統合で用いられる他，ビッグデータ解析 [12] における基本的な演算となっている．そのため，秘密計算におけるソーティングの高速化は重要な課題である．

秘密計算を実現する方法の内，複数の計算者（パーティ）が情報を分散して持つ秘密分散方式 [13] では，基本的に各計算ごとに通信が必要となり，通信にかかる時間がボトルネックとなることが多い．通信にかかる時間は，通信を行うビット数を表す通信量を通信速度で割ったものと，通信回数を表すラウンド数と遅延を掛けたものにより評価される．

1.1 先行研究

秘密分散方式における秘密計算ソートについては，いくつかの手法が提案されてきた．

まず，データのランダムシャッフルを行ってから，比較プロトコルによって大小関係のみ公開しつつ比較ソートを行う手法が提案された [6]．この方針は，data-oblivious なソートであるソーティングネットワーク [2] 等を用いる方法より高速であることが確認された [8]．

その後，秘密計算において計数ソートと置換の作用を行い基数ソートを行う手法が提案された [7]．基数ソートによる手法では，通信量を多く必要とする比較プロトコルを

[1] 東京大学工学部計数工学科
Department of Mathematical Engineering and Information Physics, Faculty of Engineering, The University of Tokyo
[2] 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo
a) yoshida-yuki814@g.ecc.u-tokyo.ac.jp
b) sada@mist.i.u-tokyo.ac.jp

用いず，ラウンド数もデータ数に依存しないことから，それまでの手法より高速に実行が行えるようになった．以降も，通信量の削減による秘密計算基数ソートの高速化が行われていった [9], [10]．また，上記の手法は 3 パーティプロトコルであったが，事前計算は依頼者の助けを借りる 2 パーティの効率的な秘密計算基数ソートプロトコルが提案された [1].

1.2 本研究の成果

本研究では，五十嵐ら (2017) [10] の手法と Attrapadung et al.(2021) [1] の手法を組み合わせて，基数ソートのサブプロトコルである L ビット整数 N 個の計数ソートの効率化を行う．具体的には，入力が与えられた後の通信量であるオンライン通信量について，既存手法では $O(2^L N \log N)$ ビットであったが，提案手法では $O(NL)$ ビットに削減する．そして， [1] の手法をベースとして依頼者の助けは借りない 3 パーティでの基数ソートプロトコルを構築し，オンライン通信量が改善することを示す．

また，本稿にて提案手法の安全性の厳密な証明は行わないが，証明の際に用いた十分条件について説明する．semi-honest な高々 1 人の攻撃者を仮定した下での十分条件は，他パーティの出力が関わらない部分と，出力がシェアである場合は容易に満たせる部分からなる．これにより，他パーティの出力も含めた同時分布の議論を行う必要がなく，証明の簡潔化が可能である．

2. 定義

既存手法及び提案手法の説明に必要な定義を行う．

2.1 表記

以降では 2 パーティないし 3 パーティプロトコルを考え，各パーティを $P_i\,(i=1,2,3)$ と表記する．また，対象のデータ数 N と素数 $p > \max(N,3)$ を固定し，p の表現ビット数を $\lambda(\sim \log_2 N)$ とする．

ある集合 S から一様ランダムに要素 e を選ぶことを $e \xleftarrow{\mathrm{R}} S$ と表記する．

$[x]$ は x の加法的シェア，$[\![x]\!]$ は x の Shamir のシェア [11] を表し，$[x]_i$ は P_i が持つシェアとする．特に，プロトコルの入出力の $P_1, P_2 : [x]_i$ は，P_i が $[x]_i$ を持つことを意味する．また，$x = (x_1, \ldots, x_N)$ に対して $[x]_i = ([x_1]_i, \ldots, [x_N]_i)$ とする．加法的シェアを $\bmod 2$ で考える場合は $[\cdot]^B$ と表記し，Shamir のシェアは $\mathbb{F}_p$ 上で考える．

あるシェアを持つパーティ数を強調する場合，2 パーティなら $[\cdot]_{12}$，3 パーティなら $[\cdot]_{123}$ と表記する．

次に，$x \in \mathbb{Z}^N, \pi \in S_N$ (S_N は N 要素の置換全体) に対し，

$$x \cdot \pi \coloneqq (x_{\pi(1)}, \ldots, x_{\pi(N)}) \tag{1}$$

と定義する．$\sigma \in S_N$ が $\sigma(i) < \sigma(j)$ ならば $(x_i, i) < (x_j, j)$ を満たす時，σ は x の安定ソートと定義する．これは，$x \cdot \sigma^{-1}$ が安定な順番でソート済である定義 [1] と同値である．置換のシェアは $[\pi] = ([\pi(1)], \ldots, [\pi(N)])$ とする．

また，$x \in \mathbb{Z}^{2^w}$, $a^w, \ldots, a^1 \in \mathbb{Z}_2$ に対し，

$$x \cdot (a^w, \ldots, a^1) \coloneqq (x_{0 \oplus a}, \ldots, x_{(2^w-1) \oplus a}) \tag{2}$$

のように XOR 置換 [1] を定義する．但し，$a = \sum_{j=1}^{w} 2^{j-1} a^j$ とし，$\oplus$ はビット毎の排他的論理和を表す．

2.2 問題設定

秘密計算基数ソートの入出力は以下で定義される．

入力 $x^1, \ldots, x^W \in \mathbb{Z}_2^N$ のシェア $[x^1]^B, \ldots, [x^W]^B$

出力 $x = \sum_{j=1}^{W} 2^{j-1} x^j$ の安定ソートのシェア $[\rho]$

ここで，パラメータ N, W は事前計算で用いてよいとする．また，以降で扱うサブプロトコルも含めて，プロトコルが出力するシェアは，取りうる値の範囲で一様ランダムであると定義する．

2.3 安全性

安全性に関して，攻撃者は semi-honest であり高々 1 人とする．この仮定の下，Canetti(2000) [4] による定義

$$\mathrm{IDEAL}_{f,\mathcal{S}} \equiv \mathrm{EXEC}_{\Pi,\mathcal{A}} \tag{3}$$

によって，プロトコル Π が安全に関数 f の計算を行うことを示す．但し，式 (3) の左辺は ideal モデルで攻撃者 $\mathcal{S}$ が得る値と f の値の同時分布，右辺は real-life モデルで攻撃者 $\mathcal{A}$ が得る値と Π の出力値の同時分布であり，等号は確率分布として等しいことを表す．以降でも，ideal/real-life モデルという言葉は [4] と同様の意味で用いる．

3. 既存手法

秘密計算基数ソートにおける既存手法について，提案手法に大きく関わる部分を説明する．

3.1 計数ソートの方針

既存手法及び提案手法では，次の手順で L ビット整数の計数ソートを行う．途中の各変数はシェアとして扱う．

a) $x \in \mathbb{Z}_{2^L}^N$ の各要素 x_i について，

$$e_k^{(x_i)} = \delta_k^{x_i} \qquad (0 \le k \le 2^L - 1) \tag{4}$$

を満たす単位ベクトル $e^{(x_i)} \in \mathbb{F}_p^{2^L}$ を生成．但し，$\delta_k^{x_i}$ は $k = x_i$ の時に 1，それ以外で 0 となる変数．

b) 次の $(f_{k,i})_{0 \le k \le 2^L-1,\, 1 \le i \le N}$ を計算する．

$$f_{k,i} = \#\{j \mid x_j < k\} + \#\{j \le i \mid x_j = k\} \tag{5}$$

$$= \sum_{l=0}^{k-1} \sum_{m=1}^{N} e_l^{(x_m)} + \sum_{m=1}^{i} e_k^{(x_m)}. \tag{6}$$

c) $\sigma(i) \coloneqq f_{x_i,i}$ と定義すると，σ は x の安定ソート．

ステップ b は加算のみで行えるため，線形性を持つシェア

ならばローカルに計算できる．よって，ステップ a 及び c の効率化が重要である．以降では，オンライン通信量の面で効率的にこれらを実現する既存手法について説明する．

3.2 単位ベクトルの生成 [1]

Attrapadung et al.(2021) [1] は，$x^L,\dots,x^1 \in \mathbb{Z}_2^N$ の加法的シェアから，$x=\sum_{j=1}^{L} 2^{j-1}x^j$ の各要素の単位ベクトルを生成する手法（プロトコル 1）を提案した．プロトコル中の式 (7) では，$p=\sum_{j=1}^{L} 2^{j-1}p^j = x \oplus c$ であるので，XOR 置換によって $e^{(c)}$ から $e^{(c\oplus p)}=e^{(x)}$ に変化する．

ステップ 1 は依頼者の助けを借りた事前計算であるので，オンライン通信量はステップ 2 の L ビットのみである．

プロトコル 1 単位ベクトルのシェアの生成 [1]

入力 $x^L,\dots,x^1 \in \mathbb{Z}_2$ の加法的シェア $[x^L]^B,\dots,[x^1]^B$.
出力 $x=\sum_{j=1}^{L} 2^{j-1}x^j \in \mathbb{Z}_{2^L}$ の単位ベクトルの加法的シェア $[e^{(x)}]$.

1: 依頼者は $c \in \mathbb{Z}_{2^L}$ を一様ランダムに生成し，ビット毎の加法的シェア $[c^L]^B,\dots,[c^1]^B$ $(c=\sum_{j=1}^{L} 2^{j-1}c^j)$ と単位ベクトルのシェア $[e^{(c)}]$ を各パーティに共有する．
2: 各パーティ P_i は $[p^j]_i \coloneqq [x^j]_i \oplus [c^j]_i\,(j=1,\dots,L)$ を計算し，シェアを共有することで $p^L,\dots,p^1 \in \mathbb{Z}_2$ を復元する．
3: 各パーティ P_i は次によって $[e^{(x)}]$ を計算する．

$$[e^{(x)}]_i \leftarrow [e^{(c)}]_i \cdot (p^L,\dots,p^1). \tag{7}$$

3.3 内積演算の効率化による置換の計算 [10]

五十嵐ら (2017) [10] は，(2,2) 出力積和（プロトコル 2）を用いて効率的な計数ソートの計算を行った．具体的には，$n=2^L$ として $a=e^{(x_i)}$ と $b=((f_{0,i}),\dots,(f_{2^L-1,i}))$ を入力すれば，出力は $f_{x_i,i}=\sigma(i)$ となる．

ステップ 3 については，(2,3)-しきい値法の Shamir のシェアが一次式からなるため乗算 1 回で二次式となることと，Shamir のシェアの復元は各シェアの線形結合により行えることを用いている．

オンライン通信量は，ステップ 4 における λ のみである．

プロトコル 2 (2,2) 出力積和 [10]

入力 (2,3)-しきい値の Shamir のシェア $[\![a]\!]_{123}, [\![b]\!]_{123}\,(a,b \in \mathbb{F}_p^n)$.
出力 加法的シェア $[c]_{12}$．但し，$c=\sum_{i=1}^{n} a_i b_i$ とする．

1: P_1, P_3 は乱数 $r \in \mathbb{F}_p$ を共有する．
2: 各パーティ P_j は，$[\![c]\!]_j \coloneqq \sum_{i=1}^{n} [\![a_i]\!]_j [\![b_i]\!]_j$ を計算する．
3: $[\![c]\!]_j$ は (3,3)-しきい値の Shamir のシェアであるので，各 P_j はある係数 λ_j を用いて加法的シェア $[c]_j=\lambda_j [\![c]\!]_j$ を計算出来る．
4: P_3 は P_2 に $c_3' \coloneqq [c]_3 - r$ を送信する．
5: P_1 は $[c]_1 + r$ を，P_2 は $[c]_2 + c_3'$ を出力する．

3.4 Double Share による置換の作用 [1]

Attrapadung et al.(2021) [1] は，置換 π に対し次を満たす組 $(\langle\pi\rangle, a, a')$ を依頼者が用意することで，パーティが π を未知としてシェアに作用させる方法を提案した．但し，各変数は条件を満たす下で一様ランダムに分布する．特に，$\langle\pi\rangle$（Double Share と呼ぶ）を固定した下で，a_1, a_2, a_1', a_2' の 4 つ中 3 つを独立一様ランダムに取ることが出来て，残り 1 つが一意に定まる．以降では，これらの制約を満たす乱数の組を LGA 乱数（[1] では *"correlated randomness for LGA"*）と呼ぶ．

$$\langle\pi\rangle = (\langle\pi\rangle_1, \langle\pi\rangle_2) = ((\pi_1,\pi_1'),(\pi_2,\pi_2')), \tag{8}$$

$$\pi = \pi_1 \circ \pi_2' = \pi_2 \circ \pi_1', \tag{9}$$

$$(a,a') = ((a_1,a_2),(a_1',a_2')), \tag{10}$$

$$a_1' + a_2' = a_1 \cdot \pi_2' + a_2 \cdot \pi_1'. \tag{11}$$

$P_i\,(i=1,2)$ が $(\langle\pi\rangle_i, a_i, a_i')$ を用いることで，Π_{Shuffle}（プロトコル 3）によりシェアに π を作用できる．

式 (9), (11) を用いて $[y]_1+[y]_2$ を変形することで正当性が確かめられる．但し，次式の成立に注意する．

$$v \in \mathbb{Z}^N, \sigma, \sigma' \in S_N, \quad (v\cdot\sigma)\cdot\sigma' = v\cdot(\sigma\circ\sigma'). \tag{12}$$

ステップ 2 において，ベクトルの表現ビット数に等しい分のオンライン通信量が必要である．

プロトコル 3 Π_{Shuffle}（依頼者 $+2$ パーティ）[1]

入力 P_1, P_2：$[x]_i, \langle\pi\rangle_i \quad (x \in \mathbb{Z}_q^N,\ \pi \in S_N,\ q$ は正整数).
出力 P_1, P_2：$[y]_i \quad (y = x\cdot\pi)$.

1: 各 P_i は LGA 乱数 $(\langle\pi\rangle_i, a_i, a_i')$ を依頼者から受け取る．但し，$a_i, a_i' \in \mathbb{Z}_q^N$ とする．
2: 各 $P_i\,(i=1,2)$ は $v_i \leftarrow [x]_i \cdot \pi_i + a_i$ を計算し，P_{i+1} に送信．
3: 各 P_i は $[y]_i \leftarrow v_{i+1}\cdot\pi_i' - a_i'$ を出力する．

3.5 基数ソートの構築 [1]

Attrapadung et al.(2021) [1] は，Shuffle と GenBitsPerm（計数ソート）をサブプロトコルとして，基数ソートプロトコル（プロトコル 4）を構築した．但し，GenBitsPerm は $v^L,\dots,v^1 \in \mathbb{Z}_2^N$ と $\gamma \in S_N$ を受け取り $(\sum_{j=1}^{L} 2^{j-1}v^j)\cdot\gamma^{-1}$ の安定ソートのシェアを得るプロトコルである．

正当性は，次の 3 点に注目した帰納法により示せる．但し，ρ_k は x の下位 Lk ビットの安定ソートとする．

1) $\gamma_k = \rho_{k-1}\circ\pi_k$ である．
2) σ_k は，$x^{[L(k-1)+1,Lk]}\cdot\rho_{k-1}^{-1}$ の安定ソートである．
3) $\delta_k = \rho_k \circ \pi_k$ である．

特に，性質 2 と基数ソート自体の正当性より $\rho_k = \sigma_k \circ \rho_{k-1}$ が成立する点が重要である．

オンライン通信量の詳細はここでは省略する．

4. 提案手法

提案手法の秘密計算基数ソートについて説明する．以降でパーティを表す添字 i を用いる時の $i+1$ については，

プロトコル 4 $\Pi_{\text{RadixSort}}$（依頼者 +2 パーティ）[1]

入力 P_1, P_2：$[x^1]_i^B, \ldots, [x^W]_i^B$ $(x^1, \ldots, x^W \in \mathbb{Z}_2^N)$.
出力 P_1, P_2：$[\rho]_i$ (ρ は $x = \sum_{j=1}^W 2^{j-1}x^j$ の安定ソート).
1: 依頼者は $k = 2, \ldots, \lceil \frac{W}{L} \rceil$ に対してランダム置換 π_k を生成し，Double Share $\{\langle\pi\rangle_k\}_{k=2,\ldots,\lceil \frac{W}{L} \rceil}$ と $\{\langle\pi_{k-1}^{-1} \circ \pi_k\rangle\}_{k=3,\ldots,\lceil \frac{W}{L} \rceil+1}$ を P_1, P_2 に送信する（$\pi_{\lceil \frac{W}{L} \rceil+1} = \mathrm{id}$, $\langle\pi_{2-1}^{-1} \circ \pi_2\rangle = \langle\pi_2\rangle$）.
2: $[\delta_1] \leftarrow \mathrm{GenBitsPerm}([x^L]^B, \ldots, [x^1]^B, \mathrm{id})$ を得る.
3: $k = 2, 3, \ldots, \lceil \frac{W}{L} \rceil$ の順に次を行う.
(a): $[b^j]^B \leftarrow \mathrm{Shuffle}([x^j]^B; \langle\pi_k\rangle)$ $(j = L(k-1)+1, \ldots, Lk)$.
(b): $[\gamma_k] \leftarrow \mathrm{Shuffle}([\delta_{k-1}]; \langle\pi_{k-1}^{-1} \circ \pi_k\rangle)$.
(c): P_i $(i = 1, 2)$ は P_{i+1} に $[\gamma_k]_i$ を送信し，γ_k を復元する.
(d): $[\sigma_k] \leftarrow \mathrm{GenBitsPerm}([b^{Lk}]^B, \ldots, [b^{L(k-1)+1}]^B, \gamma_k)$.
(e): $[\delta_k] \leftarrow [\sigma_k] \cdot \gamma_k$.
4: $[\rho] \leftarrow \mathrm{Shuffle}([\delta_{\lceil \frac{W}{L} \rceil}]; \langle\pi_{\lceil \frac{W}{L} \rceil}^{-1}\rangle)$ を出力する.

$i = 1, 2$ の場合は $2+1$ を 1 として考え，$i = 1, 2, 3$ の場合は $3+1$ を 1 として考えることとする．また，$[\cdot]^B$ ではない加法的シェアは全て $\mathbb{F}_p$ 上とする.

4.1 改善の方針

提案手法の構築で行うことは次の通りである.

1) 第 3.1 節で述べたステップ a 及び c について，別々に用いられていた効率化の既存手法を同時に用いることで，計数ソートのオンライン通信量を削減する.
2) 1) を素直に行うと，3 パーティに加えて事前計算で依頼者の助けが必要となるが，LGA 乱数などを 3 パーティで生成するプロトコルを構築し，完全な 3 パーティプロトコルとして基数ソートの構築を行う.
3) (2,2) 出力積和（プロトコル 2）の通信と Shuffle を兼ねることで，オンライン通信量のさらなる削減を行う.

これらを踏まえて構築した提案手法のプロトコルが，プロトコル 5 からプロトコル 10 である．具体的には，(1) はプロトコル 9 で，(2) は-prep の付くプロトコルで，(3) はプロトコル 7 でそれぞれ行っている．正当性や既存手法との関係性などの詳細は，次節以降で説明する.

4.2 事前計算

事前計算に当たるプロトコル，すなわち基数ソートの入力が与えられる前に全ての通信を行えるプロトコルは，LGArand-prep（プロトコル 5），LGA3-prep（プロトコル 6），GBP-prep（プロトコル 8）の 3 つである.

LGArand-prep と LGA3-prep は，依頼者の助けを借りずに LGA 乱数を生成するためのプロトコルである．簡単には，P_3 が生成した LGA 乱数を P_1, P_2 がランダムに作り替えるという構造となっている．但し，LGArand-prep における出力 d はプロトコルの安全性のために必要な出力であり，プロトコル中で用いることが目的ではない.

LGArand-prep の正当性は，LGA 乱数にて作り替えることが可能な要素と一対一対応するように乱数が生成されていることから分かる．また，LGA3-prep についても，出力の一様ランダム性は LGArand-prep により保証され，出力が正しいことも容易に確かめられる.

プロトコル 5 $\Pi_{\text{LGArand-prep}}$

入力 P_1, P_2：LGA 乱数 $(\langle\pi\rangle_i, a_i, a_i')$ $(a_i, a_i' \in \mathbb{Z}_q^N)$.
出力 P_1, P_2：LGA 乱数 $(\langle\sigma\rangle_i, b_i, b_i')$，ベクトル $d \in \mathbb{Z}_q^N$ $(\sigma = \pi, d = b_1' - a_1' - b_2 \cdot \sigma_1' + a_2 \cdot \pi_1')$.
1: $\delta_i \xleftarrow{R} S_N, c_i \xleftarrow{R} \mathbb{Z}_q^N$ $(i = 1, 2), d \xleftarrow{R} \mathbb{Z}_q^N$ を P_1, P_2 間で共有する.
2: 各 P_i は $(\sigma_i, \sigma_i') \leftarrow (\pi_i \circ \delta_i, \delta_{i+1}^{-1} \circ \pi_i')$ を計算.
3: 各 P_i は $(b_i, b_i') \leftarrow ((a_i + c_i) \cdot \delta_i, a_i' + c_{i+1} \cdot \pi_i' \pm d)$ を計算.
4: 各 P_i は $(\langle\sigma\rangle_i, b_i, b_i')$ と d を出力する.

プロトコル 6 $\Pi_{\text{LGA3-prep}}$

入力 P_1, P_2：置換 π^L, π^R,
P_3：置換 π^M.
出力 P_1, P_2：LGA 乱数 $(\langle\sigma\rangle_i, b_i, b_i')$ $(\sigma = \pi^L \circ \pi^M \circ \pi^R)$.
1: P_3 は LGA 乱数 $(\langle\pi^M\rangle_i, a_i, a_i')_{i=1,2}$ を一様ランダムに生成し，各 P_i に送信する.
2: P_i $(i = 1, 2)$ は $(\pi_i, \pi_i') \leftarrow (\pi^L \circ (\pi^M)_i, (\pi^M)_i' \circ \pi^R)$ を計算.
3: P_i $(i = 1, 2)$ は $(a_i, a_i') \leftarrow (a_i, a_i' \cdot \pi^R)$ へと更新する.
4: P_i $(i = 1, 2)$ は $((\langle\sigma\rangle_i, b_i, b_i'), d) \leftarrow \text{LGArand-prep}(\langle\pi\rangle, a, a')$ を受け取る.
5: P_i $(i = 1, 2)$ は $(\langle\sigma\rangle_i, b_i, b_i')$ を出力する.

GBP-prep は，乱数に応じた単位ベクトルを事前計算で生成するプロトコルであり，[1] の手法（プロトコル 1）とは異なり Shamir のシェアとして生成している．正当性は，$c^j = b^j \oplus a^j$ よりステップ 4 の計算が正しいため，確かめられる.

事前計算での通信量であるオフライン通信量は，GBP-prep において $O(2^L\lambda)$ ビット必要であり，L がある程度大きいならばこの項が主要項となる.

4.3 3 パーティ入力のシャッフル

3 パーティが持つ加法的シェア $[\delta]_{123}$ を入力とした場合に，Double Share を用いてシャッフルを行うプロトコルが ShuffleFrom3（プロトコル 7）である．これにより，計数ソートの結果を 3 パーティ分の加法的シェアで受け取る今回のプロトコルを効率的に処理できる.

プロトコルの内容は [1] での 2 パーティプロトコル（プロトコル 3）の場合とほとんど変わらないため，主な変更点をプロトコル 7 にて太字で示す．正当性についても，$v_1 + v_3$ の値は 2 パーティプロトコルの場合の P_1 の送信値と同様であることから分かる.

4.4 計数ソート

GenBitsPerm（プロトコル 9）は，ステップ 2(d) までは [1] の手法を，ステップ 2(e) 及び 3 では [10] の手法を用いている．ステップ 2(d) 以外についての正当性は第 3 章にて述べた通りであり，ステップ 2(d) によって γ^{-1} が正しく作用していることが分かる．よって，このプロトコルは正当である.

プロトコル 7 $\Pi_{\text{ShuffleFrom3}}$

入力 $P_1, P_2, P_3 : [\delta]_i \quad (\delta \in S_N)$.
P_1, P_2：置換 π^L, π^R.
P_3：置換 π^M.
出力 $P_1, P_2 : [\gamma]_i \quad (\gamma = \delta \circ (\pi^L \circ \pi^M \circ \pi^R))$.
1: $r \xleftarrow{\mathrm{R}} \mathbb{Z}_p^N$ **を** P_1, P_3 **間で共有する．**
2: $(\langle\sigma\rangle_i, b_i, b_i') \leftarrow \text{LGA3-prep}((\pi^L, \pi^R), \pi^M)$ **を** $P_i\,(i = 1, 2)$ **が受け取る．**
3: P_1 **は** P_3 **に** σ_1 **を送信する．**
4: $P_i\,(i = 1, 3)$ **は** $[\delta]_i \leftarrow [\delta]_i \pm r$ **へと更新する．**
5: P_3 は P_2 に $v_3 \leftarrow [\delta]_3 \cdot \sigma_1$ を送信する．
6: $P_i\,(i = 1, 2)$ は $v_i = [\delta]_i \cdot \sigma_i + b_i$ を P_{3-i} へ送信する．
7: P_1 は $[\gamma]_1 \leftarrow v_2 \cdot \sigma_1' - b_1'$ を，P_2 は $[\gamma]_2 \leftarrow (v_1 + v_3) \cdot \sigma_2' - b_2'$ を出力する．

プロトコル 8 $\Pi_{\text{GBP-prep}}$

入力 $P_1, P_2 : a^1, \ldots, a^L \in \mathbb{Z}_2$.
$P_3 : b^1, \ldots, b^L \in \mathbb{Z}_2$.
出力 $P_1, P_2 : [c^1]_i^B, \ldots, [c^L]_i^B \quad (c^j = a^j \oplus b^j)$.
P_1, P_2, P_3：(2,3)-しきい値 Shamir のシェア $[\![e^{(c)}]\!]_i$ $(c = \sum_{j=1}^L 2^{j-1} c^j)$.
1: $r^j \xleftarrow{\mathrm{R}} \mathbb{Z}_2\,(j = 1, \ldots, L), n_{1,k} \xleftarrow{\mathrm{R}} \mathbb{Z}_p\,(k = 0, \ldots, 2^L - 1), s \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{2^L}$ を P_1, P_2 間で共有する．また，$[\![0]\!]_{123} = (n_{1,k}, n_{2,k}, n_{3,k})$ が (2,3)-しきい値の Shamir のシェアとなる一意な値を取り，第 k 成分が $n_{i,k}$ であるベクトルを $n_i \in \mathbb{Z}_p^{2^L}\,(i = 1, 2, 3)$ と表す．
2: P_3 は $[b^j]_{12}^B$ と $[\![e^{(b)}]\!]_{123}$ を生成し，対応するシェアを $P_i\,(i = 1, 2)$ に送信する．また，$v_1 \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{2^L}, v_2 = [\![e^{(b)}]\!]_3 - v_1$ を生成し，v_i を $P_i\,(i = 1, 2)$ に送信する．
3: P_1 は $[c^j]_1^B = [b^j]_1^B \oplus r^j$ を，P_2 は $[c^j]_2^B = [b^j]_2^B \oplus a^j \oplus r^j$ を計算する．
4: $P_i\,(i = 1, 2)$ は $[\![e^{(c)}]\!]_i = [\![e^{(b)}]\!]_i \cdot (a^L, \ldots, a^1) + n_i$ を計算する．
5: P_1 は $w_1 = v_1 \cdot (a^L, \ldots, a^1) + n_3 + s$ を，P_2 は $w_2 = v_2 \cdot (a^L, \ldots, a^1) - s$ を P_3 に送信する．
6: P_1, P_2 は $[c^j]_i^B, [\![e^{(c)}]\!]_i$ を出力し，P_3 は $[\![e^{(c)}]\!]_3 \leftarrow w_1 + w_2$ を出力する．

オンライン通信量は，ステップ 2(c) における p_i^j の復元のみであり，合計で NL ビットである．また，この通信は γ について分かっていなくても行えることに注意する．

4.5 基数ソート

提案手法の RadixSort（プロトコル 10）では，一部プロトコルの仕様を 3 パーティに変更した以外はほとんどプロトコル 4 [1] と同じであるため，主な変更点を太字で強調した．

正当性については，$\pi_k := \pi_k^L \circ \pi_k^R$ により定義することで，プロトコル 4 の場合と同様に示すことが出来るため，ここでは省略する．

オンライン通信量については，ステップ 2 及び 3(d) で行う計数ソート，ステップ 3(a) で行う対象のデータのシャッフル，ステップ 3(b) 及び 4 で行う置換のシャッフル，ステップ 3(c) の復元の合計 4 か所で発生する．Shuffle でも ShuffleFrom3 でも，オンライン通信量は置換を作用させる対象の表現ビット数であるので，合計のオンライン通信量は

プロトコル 9 $\Pi_{\text{GenBitsPerm}}$

入力 $P_1, P_2 : [x^1]_i^B, \ldots, [x^L]_i^B \quad (x^1, \ldots, x^L \in \mathbb{Z}_2^N)$.
$P_1, P_2, P_3 : \gamma \in S_N$
出力 $P_1, P_2, P_3 : [\sigma]_i \quad (\sigma$ は $(\sum_{j=1}^L 2^{j-1} x^j) \cdot \gamma^{-1}$ の安定ソート$)$.
1: $P_i, P_{i+1}\,(i = 1, 2, 3)$ は $r_{i,i+1} \xleftarrow{\mathrm{R}} \mathbb{Z}_p^N$ を共有する．
2: $i = 1, 2, \ldots, N$ に対して，次の計算を行う．(a) から (d) においては $j = 1, \ldots, L$ とする．
(a): P_1, P_2 は $a_i^j \xleftarrow{\mathrm{R}} \mathbb{Z}_2$ を共有し，P_3 は $b_i^j \xleftarrow{\mathrm{R}} \mathbb{Z}_2$ を生成する．
(b): $([c_i^j]_{12}^B, [\![e^{(c_i)}]\!]_{123}) \leftarrow \text{GBP-prep}(a_i^j, b_i^j)$ を得る．
(c): P_1 は $[x_i^j]_1^B \oplus [c_i^j]_1^B$ を，P_2 は $[x_i^j]_2^B \oplus [c_i^j]_2^B$ を他の 2 パーティへ送信し，3 パーティは $p_i^j = x_i^j \oplus c_i^j$ を共有する．
(d): 3 パーティは，p_i^j 及び $[\![e^{(c_i)}]\!]$ について，添字 i に対し置換 γ^{-1} を作用させる次のような更新を行う．

$$(p_1^j, p_2^j, \ldots, p_N^j) \leftarrow (p_1^j, \ldots, p_N^j) \cdot \gamma^{-1} \tag{13}$$
$$= (p_{\gamma^{-1}(1)}^j, \ldots, p_{\gamma^{-1}(N)}^j) \tag{14}$$
$$([\![e^{(c_1)}]\!], \ldots, [\![e^{(c_N)}]\!]) \leftarrow ([\![e^{(c_i)}]\!])_{1 \le i \le N} \cdot \gamma^{-1} \tag{15}$$
$$= ([\![e^{(c_{\gamma^{-1}(i)})}]\!])_{1 \le i \le N} \tag{16}$$

(e): 3 パーティは次をローカル計算する．但し，$[\![\sigma_i]\!]_{123}$ は (3,3)-しきい値 Shamir のシェアとなることに注意する．

$$[\![e^{(x_i)}]\!] \leftarrow [\![e^{(c_i)}]\!] \cdot (p_i^L, \ldots, p_i^1) \tag{17}$$
$$[\![f_{k,i}]\!] \leftarrow \#\{j \mid x_j < k\} + \#\{j \le i \mid x_j = k\} \tag{18}$$
$$= \sum_{l=0}^{k-1} \sum_{m=1}^{N} [\![(e^{(x_m)})_l]\!] + \sum_{m=1}^{i} [\![(e^{(x_m)})_k]\!] \tag{19}$$
$$(k = 0, \ldots, 2^L - 1) \tag{20}$$
$$[\![\sigma_i]\!] \leftarrow [\![f_{x_i,i}]\!] \tag{21}$$
$$= \sum_{k=0}^{2^L-1} [\![(e^{(x_i)})_k]\!][\![f_{k,i}]\!] \tag{22}$$

3: (3,3)-しきい値 Shamir のシェア $[\![\sigma]\!]_{123}$ をローカルに加法的シェア $[\sigma]_{123}$ に変換する．そして，$P_i\,(i = 1, 2, 3)$ は $[\sigma]_i \leftarrow [\sigma]_i + r_{i,i+1} - r_{i-1,i}$ へと更新し，$[\sigma]_i$ を出力する．

$$NL + \left(\left\lceil \frac{W}{L} \right\rceil - 1\right)(2NL + 2N\lambda) + N\lambda \tag{23}$$
$$= 2N \left\lceil \frac{W}{L} \right\rceil (L + \lambda) \tag{24}$$

である．W が L の倍数だとすれば $2NW(1 + \lambda/L)$ となるため，主要項 $NW\lambda$ の係数は $2/L$ となる．

オンライン通信のラウンド数については，ステップ 2 及び 4 での 1 回と，ステップ 3 では

- (a) と (b) が同時に通信可能
- (c) と (d) が同時に通信可能
  （GenBitsPerm で確認したように，プロトコル内部のオンライン通信量は γ を用いないで可能）

となっていることから，合計で $2\lceil \frac{W}{L} \rceil$ 回である．

5. 性能比較

残念ながら，提案手法を実装することが出来ていないため，実際の速度を測ることは出来ていない．しかし，最も実行時間に影響するのは通信の量及び回数である．ここでは，前章で述べた提案手法の通信量について，既存の秘密

プロトコル 10 $\Pi_{\text{RadixSort}}$（3 パーティ）

入力 P_1, P_2：$[x^1]_i^B, \ldots, [x^W]_i^B$ $(x^1, \ldots, x^W \in \mathbb{Z}_2^N)$.
出力 P_1, P_2：$[\rho]_i$ （ρ は $x = \sum_{j=1}^{W} 2^{j-1} x^j$ の安定ソート）.

1: $k = 2, \ldots, \lceil \frac{W}{L} \rceil$ **に対して，** P_1, P_2 **は** $\pi_k^R \xleftarrow{R} S_N$ **を共有し，** P_3 **は** $\pi_k^L \xleftarrow{R} S_N$ **を生成する．**
また， $k = 1, \lceil \frac{W}{L} \rceil + 1$ **に対し** $\pi_k^R = \pi_k^L = \mathrm{id}$ **と定義する．**
2: $[\delta_1]_{\mathbf{123}} \leftarrow \mathrm{GenBitsPerm}([x^L]^B, \ldots, [x^1]^B)$ を得る．
3: $k = 2, 3, \ldots, \lceil \frac{W}{L} \rceil$ の順に次を行う．
(a): $[b^j]_{12}^B \leftarrow \mathrm{Shuffle}([x^j]^B; (\mathrm{id}, \pi_k^R), \pi_k^L)$ $(j = L(k-1)+1, \ldots, Lk)$．**但し Shuffle は，** Π_{Shuffle} **（プロトコル 3）におけるステップ 1 を LGA3-prep（プロトコル 6）に変更したプロトコルとする．**
(b): $[\gamma_k]_{12} \leftarrow \mathrm{ShuffleFrom3}([\delta_{k-1}]; ((\pi_{k-1}^R)^{-1}, \pi_k^R), (\pi_{k-1}^L)^{-1} \circ \pi_k^L)$ **を計算する．**
(c): P_i $(i = 1, 2)$ が $[\gamma_k]_i$ を他の 2 パーティに送信し，**3 パーティ**は γ_k を共有する．
(d): $[\sigma_k]_{\mathbf{123}} \leftarrow \mathrm{GenBitsPerm}([b^{Lk}]^B, \ldots, [b^{L(k-1)+1}]^B, \gamma_k)$.
(e): $[\delta_k]_{\mathbf{123}} \leftarrow [\sigma_k] \cdot \gamma_k$.
4: $[\rho]_{12} \leftarrow \mathrm{ShuffleFrom3}([\delta_{\lceil \frac{W}{L} \rceil}]; ((\pi_{\lceil \frac{W}{L} \rceil}^R)^{-1}, \mathrm{id}), (\pi_{\lceil \frac{W}{L} \rceil}^L)^{-1})$ **を計算し，出力する．**

表 1 IHKC ソート [10], AHM+ソート [1], 提案手法の通信量（データ 1 件当たり）のオーダー比較
Table 1 Order comparison of communication (per data) with IHKC sort [10], AHM+ sort [1], and ours

手法	$m(m')$	通信量	
		オフライン	オンライン
IHKC [10]	3(3)	$O(\frac{2^L}{L} W \log N)^*$	$O(\frac{2^L}{L} W \log N)$
AHM+ [1]	2(3)	$O(\frac{2^L}{L} W \log N)$	$O(\frac{2^L}{L} W \log N)$
提案手法	3(3)	$O(\frac{2^L}{L} W \log N)$	$O(W(1 + \frac{\log N}{L}))$

*Beaver Triple [3] 等を事前計算して乗算を行うと仮定

計算ソートでも高速な手法 [1], [10] との比較を行う．

5.1 L を変数とした場合のオーダー評価

L を変数として見た時の通信量のオーダーを既存手法と比較した結果が**表 1** である．但し，パーティ数はオンライン m 人，オフライン m' 人（依頼者含む事前計算の人数）ならば $m(m')$ と表記する．提案手法のオフライン通信量に関しては，データ 1 件当たり GBP-prep を $\lceil \frac{W}{L} \rceil$ 回呼び出すことから計算できる．

表 1 より，オフライン通信量は既存手法から変化しないが，オンライン通信量は提案手法のみ L に対して減少することが確認できる．特に，提案手法は $W/L = o(W)$ の下でオンライン通信量 $o(W \log N)$ を達成できて，$W/L = O(1)$ ならば，膨大な事前計算及びローカル計算は必要となるがオンライン通信量 $O(W + \log N)$ を達成できる．

5.2 L を定数とした場合

L を定数とした場合のオンライン通信量の主要項 $W\lambda$ の係数部分について，既存手法と提案手法を比較したものが**表 2** である．但し，各論文で扱っていない L の場合の通信量は – により省略した．

表 2 IHKC ソート [10]，AHM+ ソート [1]，提案手法の通信量（データ 1 件当たり）とラウンド数の比較
Table 2 Comparison of communication (per data) and rounds with IHKC sort [10], AHM+ sort [1], and ours

手法	オンライン通信量			ラウンド数
	$L=2$	$L=3$	$L=4$	オンライン
IHKC [10]	$2.16W\lambda^*$	–	–	$5\lceil \frac{W}{L} \rceil - 2$
AHM+ [1]	$2.5W\lambda$	$3W\lambda$	–	$3\lceil \frac{W}{L} \rceil$
提案手法	$W\lambda$	$\frac{2}{3}W\lambda$	$0.5W\lambda$	$2\lceil \frac{W}{L} \rceil$

* パーティ間通信量を平均的に評価

表 2 から分かるように，$L = 2$ 同士で比較しても，提案手法のオンライン通信量は既存手法の 0.4 倍から 0.5 倍程度となっており．大きく改善している．また，$L = 3, 4$ と増加させることで，より大きな改善となる．

提案手法のラウンド数については，同じ L で比べた場合でも IHKC ソートの約 0.4 倍，AHM+ソートの約 0.67 倍のラウンド数となっている．ラウンド数による遅延は，N が大きい場合はほとんど実行時間に影響しないが，N が小さく遅延の大きい環境であれば影響が出てくるため，この改善も重要であると考えられる．

6. 安全性

本章では，安全性の定義について一般的な定義を説明した後，今回のプロトコルの安全性の証明に用いた十分条件について説明する．また，証明手法の例として，Beaver Triple [3] を用いた乗算が十分条件を満たすことを示す．

6.1 仮定

攻撃者は高々 1 パーティとし，プロトコルの実行中に攻撃者が変わることはない（non-active）とする．また，攻撃者含む全パーティはプロトコルに従う（semi-honest）とする．攻撃者は，計算能力に仮定を置かずに，自身の入力，自身が生成した乱数，他のパーティからのメッセージを用いて計算を行う．そして，自分以外の 2 パーティ間の通信内容を知ることは出来ないとする（secure channels setting）．

安全性の議論を単純に考えるため，計算対象についてもいくつか仮定を置く．以降では確率的関数 f を計算するプロトコル Π を考えるが，f の定義域は有限とする．また，f の計算時間や Π の実行時間は入出力長を下限とする．

6.2 シミュレータを用いた定義 [5]

前節の仮定の下で，Goldreich(2009) [5] によるシミュレータを用いた次の定義を考える．以降ではこの定義が成立することをシミュレート可能であると表記する．ここで，入力が x の時にパーティ P_i がプロトコルの実行によって得る値を $\mathrm{view}_i^{\Pi}(x)$，プロトコルの出力を $\mathrm{output}_i^{\Pi}(x)$ とし，$\mathrm{output}^{\Pi}(x) = (\mathrm{output}_i^{\Pi}(x))_{1 \le i \le 3}$ とする．

$$\forall i \in \{1,2,3\}, \exists \text{確率的アルゴリズム } \mathrm{Sim}_i, \quad (25)$$

$$\{(\mathrm{view}_i^{\Pi}(x), \mathrm{output}^{\Pi}(x))\}_x \quad (26)$$

$$\equiv \{(\mathrm{Sim}_i(x_i, f_i(x)), f(x))\}_x \quad (27)$$

但し，[5] においては，Sim_i は入力に対して多項式時間で計算を行うという条件が存在する．

Sim_i の条件として，入力に対する多項式時間の代わりに，プロトコル Π の計算時間の多項式時間で計算を行うという条件を考える．すると，シミュレート可能ならば Canetti(2000) [4] による定義（式 (3)）を満たすことが証明できる．これは，前節の仮定を用いつつ，[5] におけるシミュレート可能の同値条件の証明と同様の手法で示せる．

すなわち，式 (3) の意味で安全であることを示すためには，式 (27) を満たす，プロトコル Π の計算時間の多項式時間で計算を行うシミュレータ Sim_i を具体的に構築すればよい．以降では，Sim_i の計算時間は Π の計算時間の多項式時間であることを前提に議論を行う．

6.3 シミュレート可能な十分条件

シミュレート可能であることは，任意の攻撃者 $\mathcal{A}$ を想定する必要がないという点で，式 (3) より示しやすい条件だと考えられる．しかし，構築したシミュレータが等式を満たすか判定する際に，攻撃者として想定しているパーティ以外の出力との同時分布を考える必要があった．

そこで本研究では，次の 3 条件を全て満たすならばシミュレート可能であることを証明した．

1) $\{\mathrm{output}^{\Pi}(x)\}_x \equiv \{f(x)\}_x$ が成立（Π の正当性）
2) 任意の x に対し次が成立（式 (27) より弱い条件）
   $(\mathrm{view}_i^{\Pi}(x), \mathrm{output}_i^{\Pi}(x)) \equiv (\mathrm{Sim}_i(x_i, f_i(x)), f_i(x))$
3) $x, \mathrm{output}_i^{\Pi}(x)$ を任意に固定した時，$\mathrm{view}_i^{\Pi}(x)$ と $(\mathrm{output}_j^{\Pi}(x))_{j\neq i}$ は独立に分布する．

この 3 条件の成立が十分条件であることの証明において本質的であるのは，次の等式の成立である．

$$\Pr[\mathrm{Sim}_i(x) = \alpha, (f_j(x))_{j\neq i} = \gamma \mid f_i(x) = \beta] \quad (28)$$

$$= \Pr[\mathrm{Sim}_i(x) = \alpha \mid f_i(x) = \beta] \quad (29)$$

$$\times \Pr[(f_j(x))_{j\neq i} = \gamma \mid f_i(x) = \beta]. \quad (30)$$

式 (30) は，real-life モデルにおける条件 3 と対応した，ideal モデルにおけるある種の独立性を表す．確率的アルゴリズム Sim_i が用いる乱数と確率的関数 f が用いる乱数を明示的に扱うことで式 (30) の成立が示せる．

6.4 証明方針

条件 3 は，出力がシェアであるプロトコルにおいて，容易に満たせることが多い．加法的シェアを出力するプロトコルを例とすると，2 パーティプロトコルであれば，$x, \mathrm{output}_i^{\Pi}(x)$ を固定した時点で $\mathrm{output}_{i+1}^{\Pi}(x)$ の値は一意であるので，条件 3 は自明に成立する．また，3 パーティプロトコルであれば，P_{i+1}, P_{i+2} のみ共有する乱数 r によって $(\mathrm{output}_j^{\Pi}(x))_{j\neq i}$ の分布を $\mathrm{view}_i^{\Pi}(x)$ から独立した分布に変更できるため，条件 3 が成立する．

プロトコル 11 Beaver Triple を用いた乗算 [3]

入力 P_1, P_2 : $[x]_i, [y]_i \quad (x, y \in \mathbb{F}_p)$.
出力 P_1, P_2 : $[z]_i \quad (z = xy)$.

1: P_i は乱数 $a, b \xleftarrow{R} \mathbb{F}_p, c = ab$ のシェア $[a]_i, [b]_i, [c]_i$ を受け取る．
2: P_i は $[\sigma]_i \coloneqq [x]_i - [a]_i, [\rho]_i \coloneqq [y]_i - [b]_i$ を計算する．
3: P_i は互いに自身のシェアを送信し，σ, ρ を復元する．
4: $[z] \coloneqq \sigma\rho + [a]\cdot\rho + \sigma\cdot[b] + [c]$ を計算し，出力する．
   但し，$\sigma\rho$ は $[z]_1$ に加算するものとする．

条件 1 の正当性もプロトコルの提案時点で多くの場合示されているので，実質的に示す必要があるのは，式 (27) より弱い条件である条件 2 のみである．条件 2 については，Sim_i が必要な量の一様ランダムな変数を生成しつつ出力を行い，変数間の関係式が等しいことなどを確認して示した．提案手法の証明は記述量が多いため本稿では省略し，Beaver Triple [3] による乗算の証明を例として，十分条件を用いる手法を説明する．

6.5 具体例

Beaver Triple を用いた乗算 [3] をプロトコル 11 に示す．ここで，ステップ 1 の乱数の組の生成の具体的手法については考えない．簡単には，依頼者に事前計算してもらう方法が考えられる．

十分条件 1 が成立する証明は省略する．また，条件 3 は，前節で述べた通り 2 パーティプロトコルが加法的シェアを出力するならば自明に成立．よって，条件 2 の成立を示す．

$$\mathrm{view}_1^{\Pi}([x],[y]) = ([x]_1, [y]_1, [a]_1, [b]_1, [c]_1, [\sigma]_2, [\rho]_2) \quad (31)$$

に対応する形で，Sim_1 は

$$\mathrm{Sim}_1([x]_1, [y]_1, [z]_1) \quad (32)$$

$$= ([x]_1, [y]_1, \alpha, \beta, \quad (33)$$

$$[z]_1 - (\gamma + [x]_1)(\delta + [y]_1) + \alpha\beta, \gamma, \delta) \quad (34)$$

と定義する．但し，$\alpha, \beta, \gamma, \delta \xleftarrow{R} \mathbb{F}_p$ とする．

$\alpha, \beta, \gamma, \delta$ の 4 項に対応する部分は $[a]_1, [b]_1, [\sigma]_2, [\rho]_2$ であり，これらはステップ 1 及び 2 での計算方法から一様ランダムかつ独立である．ゆえに，これらの項を任意に固定した下で，real-life モデルの 2 項 $([c]_1, [z]_1)$ の分布と，ideal モデルの 2 項

$$([z]_1 - (\gamma + [x]_1)(\delta + [y]_1) + \alpha\beta, [z]_1) \quad (35)$$

の分布が一致すればよい．real-life モデルでは，$[c]_1$ が一様分布に従い，その値から $[z]_1$ が一意に定まる．一方で，ideal モデルでは，シェア $[z]_1$ が定義より一様分布に従い，その値から式 (35) 第一項が一意に定まる．さらに，これらの関係式が等しいことが確かめられるので，各々のモデル

の 2 項の同時分布が等しいことが分かる．

以上より，条件 2 も成立するため，プロトコル 11 は安全にシェアの乗算を行う．

7. むすび

本研究の成果のまとめと，今後の課題について述べる．

7.1 研究成果

まず，semi-honest 攻撃者を仮定した秘密計算基数ソートについて，既存の 2 手法 [1], [10] を組み合わせることで，オンライン通信量の削減に成功した．具体的に，W ビット整数 N 個のデータを L ビット毎に分割する場合，計数ソートはオンライン通信量 NL ビット，基数ソートは $2NW(1 + (\log_2 N)/L)$ ビット程度とした．この結果を既存手法と比較すると，事前計算を既存手法と同程度に抑える $L = 2$ 程度でもオンライン通信量は半分以下であり，ラウンド数も 2/3 倍程度で大きく改善している．さらに，L に対してオンライン通信量が単調に減少する秘密計算基数ソートの手法は本研究が初である．

また，安全性の証明についての考察を行った．これにより，semi-honest 攻撃者を仮定して，シェアを出力するプロトコルの安全性の証明では，他パーティの出力分布を考慮しない条件 2 のみを示せばよいことが分かった．

7.2 今後の課題

今後の課題として主なものを説明する．

まず，提案手法を実装して速度を計測することが課題となる．これにより，既存手法（ [1], [10] はサブプロトコルを測定済）より高速に実行可能かどうかの確認や，ローカル計算量も考慮した実用上高速となる最適な L の決定などが行えると考えられる．

また，事前計算についての考察が挙げられる．今回は深く扱わなかったが，オフライン通信量のオーダーは変化していないため，擬似乱数等を用いた削減が重要だと考えられる．その場合にどの程度効率化できるか，そして安全性はどうなるかについて，考察が必要である．また，実社会では N と入力が同時に分かることも考えられ，特に S_N を用いた事前計算をオンライン化した場合の効率的処理が課題となる．

最後に，さらなるオンライン通信量の削減が考えられる．このことに関係して，膨大な事前計算なしにオンライン通信量 $o(NW \log N)$ ビットを達成するために，L ビット整数の計数ソート結果は NL ビットで表現できることを応用することが考えられる．しかし，現在の手法では S_N 全体が群をなすことを用いているため，単純に S_N の部分集合を用いることは出来ない．この点を解消する秘匿方法を考案するか，全く新たな手法を提案することが，さらなるオンライン通信量の削減のための課題であると考えられる．

謝辞 本研究にあたって，坂本比呂志先生，渋谷哲朗先生，申吉浩先生，清水佳奈先生から研究内容に関する貴重なご意見を多く頂きました．また，高木剛先生からの安全性に関するご指摘を受けて，証明に関する成果を考察するに至りました．この場をお借りして深く感謝し，お礼を申し上げます．

参考文献

[1] Attrapadung, N., Hanaoaka, G., Matsuda, T., Morita, H., Ohara, K., Schuldt, J. C. N., Teruya, T. and Tozawa, K.: Oblivious Linear Group Actions and Applications, *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, New York, NY, USA, Association for Computing Machinery, pp. 630–650 (online), DOI: 10.1145/3460120.3484584 (2021).

[2] Batcher, K. E.: Sorting networks and their applications, *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, AFIPS '68 (Spring), New York, NY, USA, Association for Computing Machinery, pp. 307–314 (online), DOI: 10.1145/1468075.1468121 (1968).

[3] Beaver, D.: Efficient Multiparty Protocols Using Circuit Randomization, *Advances in Cryptology — CRYPTO '91* (Feigenbaum, J., ed.), Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 420–432 (1992).

[4] Canetti, R.: Security and composition of multiparty cryptographic protocols, *Journal of Cryptology*, Vol. 13, No. 1, pp. 143–202 (2000).

[5] Goldreich, O.: *Foundations of cryptography: volume 2, basic applications*, Cambridge university press (2009).

[6] 濱田浩気，五十嵐大，千田浩司，高橋克巳：3 パーティ秘匿関数計算のランダム置換プロトコル，CSS2010, IPSJ (2010).

[7] 濱田浩気，五十嵐大，千田浩司，高橋克巳：秘匿関数計算上の線形時間ソート，SCIS2011, ISEC (2011).

[8] Hamada, K., Kikuchi, R., Ikarashi, D., Chida, K. and Takahashi, K.: Practically Efficient Multi-party Sorting Protocols from Comparison Sort Algorithms, *Information Security and Cryptology – ICISC 2012* (Kwon, T., Lee, M. K. and Kwon, D., eds.), Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 202–216 (2013).

[9] 五十嵐大，濱田浩気，　菊池亮，千田浩司：インターネット環境レスポンス 1 秒の統計処理を目指した，秘密計算基数ソートの改良，SCIS2014, ISEC (2014).

[10] 五十嵐大，濱田浩気，　菊池亮，千田浩司：超高速秘密計算ソートの設計と実装:秘密計算がスクリプト言語に並ぶ日，CSS2017, IPSJ (2017).

[11] Shamir, A.: How to Share a Secret, *Communications of the ACM*, Vol. 22, No. 11, pp. 612–613 (online), DOI: 10.1145/359168.359176 (1979).

[12] Vitter, J. S.: Algorithms and data structures for external memory, *Foundations and Trends in Theoretical Computer Science*, Vol. 2, No. 4, pp. 305–474 (2008).

[13] Yao, A. C.: Protocols for secure computations, *23rd Annual Symposium on Foundations of Computer Science*, pp. 160–164 (1982).