大規模環境での通信効率に優れた階層型秘密計算

杉本 航太[1,2,a)]　大原 一真[3]　渡邉 洋平[1]　岩本 貢[1]

概要：秘密計算（Multi-Party Computation: MPC）とは，複数の参加者が自身の入力を互いに秘匿したまま，所望の関数を安全に計算する暗号技術である．MPC プロトコルは一般に参加者全員が常時オンラインで互いに通信可能な状態を維持する必要があるため，参加者数の増大に伴い各参加者の通信負荷が増大するとともに，プロトコルの遂行が困難になるという課題がある．この課題に対する解決手法の一つとして，杉本らは SCIS 2025 にて，参加者を事前にグループ分けし，グループ毎に局所的に計算を行う「グループステージ」と，その結果を用いて全体の出力を得る「サーバステージ」の 2 つのステージから構成される「二段階秘密計算」を提案した．本稿では，二段階秘密計算を任意ステージ数に拡張する形で一般化した「階層型秘密計算」を提案する．また，秘密分散法に基づく具体的なプロトコル構成を行い，通信複雑性等の評価指標に関して既存方式との比較を行う．

キーワード：秘密計算, 階層構造, 大規模システム, 通信複雑性

Hierarchical Multi-Party Computation: A Scalable and Communication-Efficient Approach

Kota Sugimoto[1,2,a)]　Kazuma Ohara[3]　Yohei Watanabe[1]　Mitsugu Iwamoto[1]

Abstract: Multi-Party Computation (MPC) is a cryptographic technique that allows multiple parties to jointly compute a desired function while keeping their respective inputs private. In typical MPC protocols, all parties are required to remain online and be mutually communicable throughout the execution. Therefore, as the number of parties increases, this leads to greater communication overhead per party and makes protocol execution increasingly difficult to sustain. As a solution to this issue, Sugimoto et al. proposed at SCIS 2025 a model called "Two-Stage MPC," which consists of two stages: a group stage, where parties are pre-divided into groups and perform local computations within each group, and a server stage, where the intermediate results are aggregated to produce the whole output. In this paper, we generalize the Two-Stage MPC by allowing an arbitrary number of stages, and propose a new model called Hierarchical MPC. We also construct a concrete protocol based on secret sharing and evaluate its performance by comparing communication complexity and other metrics with existing protocols.

Keywords: Multi-party computation, hierarchical construction, large-scale system, communication complexity

1. はじめに

秘密計算（Multi-Party Computation: MPC）プロトコルとは，複数の参加者が自身の秘密情報をほかの参加者に秘匿しながら，それらを入力とする所望の関数の計算結果を出力する暗号技術である．MPC プロトコルはプライバシー保護とデータ分析の両立を実現する技術として社会実

[1] 電気通信大学
The University of Electro-Communications
[2] 国立研究開発法人 情報通信研究機構
National Institute of Information and Communications Technology
[3] 国立研究開発法人 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology
[a)] sugimoto-k@uec.ac.jp

装が期待されており，効率性や安全性の向上を目的とした多くの研究が行われてきた．従来の MPC プロトコルの多くは，プロトコルの各ラウンドにおいて全参加者が同時にオンラインで通信可能であるという「全参加者常時オンライン要件」，および各参加者が他参加者全員と直接通信できることを前提としている．しかし，この前提のもとでは，参加者数の増大に伴い各参加者の通信負荷が急増するとともに，マシントラブルやネットワーク障害によりプロトコルの完遂が困難となる．したがって，大規模環境ではこれらの要件が深刻なボトルネックとなる．

1.1 関連研究

BGW プロトコル [2] は，秘密分散法と多項式補完を用いた情報理論的安全性を満たす汎用 MPC プロトコルの古典的構成である．BGW プロトコルは安全性と汎用性が高い一方で，全参加者常時オンライン要件を満たす必要があり，また，各参加者が他の参加者全員と通信を行う必要がある．このため，参加者数が多い大規模環境では通信コストの増大やプロトコルの遂行が困難になるという課題がある．

これら「参加者数の増大に伴う通信効率の悪化」や「全参加者常時オンライン要件」の課題に対して，様々なアプローチが提案されてきた．以下に代表的な研究を整理する．

1.1.1 通信効率の改善が主眼の研究

Damgård–Nielsen プロトコル（DN プロトコル）[7] は秘密分散法に基づく汎用 MPC プロトコルであり，King モデルと「double sharings」という相関乱数を用いることで回路評価に要する通信量を参加者数の線形オーダに抑えた方式である．ATLAS [9] は DN プロトコルの改良方式であり，double sharings 生成のコスト削減により通信量を削減し，さらに回路を二層の構造に変換してから評価することでラウンド数を削減している．ただし，いずれの方式も全参加者常時オンライン要件および全員が相互に通信可能なネットワーク構造を前提としている．

Boyle らの方式 [4] は，各パーティが通信するべき他のパーティ数（communication locality）の概念を導入し，これを $\mathrm{polylog}(n)$ に抑える構成を提案している．さらに，安全なマルチ署名，完全準同型暗号（FHE）と NIZK を組み合わせ，総ラウンド数やメッセージサイズを $\mathrm{polylog}(n)$ に抑えている．一方で，計算量的安全性に依存し，かつ FHE 暗号文のメッセージ長 ℓ に比例する通信量が発生する．

1.1.2 全参加者常時オンライン要件の解消が主眼の研究

Transferrable MPC (T-MPC) [6] は，スマートグリッドのような木構造ネットワークにおいて，ある参加者集合が実行した秘密計算の結果を別の集合に安全に引き継ぐことを可能にする MPC である．T-MPC では，参加者集合が計算した結果のシェアを各ノードが受け取り，それぞれがそのシェアを再分散して次の参加者集合の全ノードに送信する．この構成により，計算は複数の部分集合間で段階的に進み，各段階では当該集合のみがオンラインであればよく，全参加者常時オンライン要件が不要となる．一方，計算可能な関数は，木構造に沿った局所関数の合成として表現できる関数に制限される．

Fluid MPC [5] は，計算の途中で参加者が自由に参加・離脱できる動的参加モデルを想定した MPC である．この方式では，計算対象の関数を複数のエポックに分割し，各エポックは全参加者集合から選ばれた少人数のグループ（committee）によって実行される．得られた結果は秘密分散の形で次の committee に再分散され，これを繰り返すことで全体の計算が進む．この仕組みにより，全員が計算全体を通じてオンラインである必要がなくなり，計算資源や利用可能時間に制約のある参加者や，計算専用ノードを用いた「MPC-as-a-Service」形態での利用が可能となる．さらに，各エポックごとに最小限の通信で計算状態を引き継げるため，長い計算や参加者が頻繁に入れ替わる環境でも効率的かつ安全に計算を進められる．ここでの「流動性（fluidity）」とは参加者の構成が変化しても計算を継続できる柔軟さを指し，Fluid MPC はその最大値を実現している．

You Only Speak Once (YOSO) [8] は，各参加者がプロトコル実行中に一度だけメッセージを送信することで計算を行う MPC のモデルであり，ブロックチェーンなどの大規模環境を想定したものである．YOSO モデルでは，関数の計算は事前にランダムに決められた committee により段階的に実行され，各ラウンドでは 1 つの committee のみが計算とメッセージ送信を担当する．この構造により，全参加者常時オンライン要件を解消すると共に，通信量の抑制を実現している．

1.1.3 両課題の解消を主眼とする研究

Non-Interactive MPC（NIMPC）[1] は，相関乱数を前処理段階で配布し，オンラインフェーズでは一切の双方向通信を行わずに関数計算を実現する MPC である．構成によっては，複数パーティからの一方向送信を集約して出力を生成するサーバ的役割を持つ集約者を用いる場合があるが，必須ではない．この方式は対称関数やしきい値関数のような特定の関数に対して非常に効率的に機能するが，完全な汎用性は持たない．

杉本らは [12] にて，参加者が事前に重複のないグループに分割されたのち，グループごとに独立に秘密計算プロトコルを実行する「グループステージ」と，グループステージでの出力を受け取った複数のサーバがそれらを入力として秘密計算プロトコルを実行し，その出力をプロトコル全体の出力として得るという「サーバステージ」という二つのステージから構成される「二段階秘密計算」を提案した．二段階秘密計算では，グループステージでは各グループが独立に MPC プロトコル実行を行うため，他のグループの参加者がオフラインの場合でも，グループ内の参加者さえ

オンライン状態であればプロトコル実行を進めることができるという点で，全参加者常時オンライン要件を解消し，同時に，各参加者の通信相手は局所的に制限することで各参加者の通信量を抑えることができる．

1.2 本研究の貢献

提案方式と関連研究との比較を表 1 に示す．各項目別の貢献を以下に示す．

通信量の抑制． 提案方式は階層的な木構造と中間結果の集約を組み合わせることで，プロトコル全体の総通信量を $\mathcal{O}\left(n^{1+1/d}\right)$ に抑えることができる（ここで n は入力を持つ参加者数，d はプロトコルのステージ数）．さらに，参加者をサイズ $n^{1/d}$ の小規模なグループに分割して計算を進める構成により，各参加者が通信しなければならない相手の数も局所的に限定されるため，各参加者の通信量も $\mathcal{O}\left(n^{1/d}\right)$ に抑えられる．

全参加者常時オンライン要件の解消． 提案方式では，グループ分けに基づく階層化により計算を段階的に進める構成をとることで，全参加者常時オンライン要件を解消した．各グループではグループ内の参加者のみがオンラインであればよく，その結果，大規模環境における長時間稼働でも現実的に秘密計算を実行できる．

既存プロトコルの活用． 提案方式は，各グループでの計算に線形秘密分散ベースの MPC プロトコルを構成要素として採用するため，既存のさまざまなプロトコルを容易に組み込むことができる．これにより，既存ライブラリの活用や，利用環境に応じたプロトコル選択，さらには将来的な改良方式の導入が可能となる．

2. 準備

2.1 記法

正の整数 n に対して $[n] \coloneqq \{1, 2, \ldots, n\}$ とし，実数 a に対して $\lfloor a \rfloor \coloneqq \max\{n \in \mathbb{Z} \mid n \leq a\}$ とする．集合 $\mathcal{S}$ に対して，$\mathcal{S}$ の要素数を $|\mathcal{S}|$ と表す．有限体 $\mathbb{F}$ 上の線形秘密分散法において，$x \in \mathbb{F}$ に対する i 番目のシェアを $[x]_i$ と表す．また，$[x]_i$ が参加者集合 $\mathcal{P}$ に分散されているシェアの i 番目であることを明示するために，$[x]_i^{(\mathcal{P})}$ と表す場合がある．

2.2 Hybrid Model [10]

Hybrid モデルとは，参加者が実際のメッセージを用いてプロトコルを実現するために，参加者が通常のメッセージ送受信に加えて，特定のサブ機能（subfunctionality）への理想的なオラクルアクセスを利用できるモデルである．サブ機能を $\mathcal{F}_{f_1}, \mathcal{F}_{f_2}, \ldots,$ としたときの hybrid モデルを $(\mathcal{F}_{f_1}, \mathcal{F}_{f_2}, \ldots)$-hybrid モデルと表す．

2.3 Secret Sharing Scheme

秘密分散法（Secret Sharing Scheme: SSS）[3], [11] とは，秘密情報を複数のシェアと呼ばれるデータに符号化し，複数の参加者に分散する手法の総称である．SSS では，定められた条件を満たすようなシェアの集合からは元の秘密情報が計算できる一方で，条件を満たさないようなシェアの集合からは元の秘密情報に関する情報が一切得られない，という性質を満たす．このとき，シェアの集合から元の秘密情報を計算する操作を「復元」という．

Linear Secret Sharing Scheme (LSSS) とは SSS の一種であり，各シェアが秘密情報と乱数の線形結合として与えられるという構造的特徴を持つ方式である．この線形性により，秘密情報に関する和やスカラー倍等の線形演算を，秘密情報の復元を行わずにシェアに対するローカルな処理で実現できる．

3. 構成要素

提案方式では，グループごとの安全な関数計算のために LSSS ベース MPC プロトコルを構成に用い，さらに特殊なシェア構造を導入することで，次のグループへの計算結果の安全な引き継ぎを実現する．本節ではまず既存の LSSS ベース MPC について確認し，その後に linked sharing および masked to shares の説明を行う．なお，本稿では安全性は標準的なシミュレーションベース安全性に従って定義する．すなわち，攻撃者集合の view の確率分布が，その入力と出力のみを与えられたシミュレータによって再現可能であるとき安全であるとする．

3.1 LSSS ベース MPC

$\mathcal{F}_{\mathsf{LSSSMPC}}^{f}$ は線形秘密分散に基づく MPC の理想機能であり，線形秘密分散法のシェアを保持している複数のパーティが，秘密値を復元することなく，秘密値を入力とした関数 f の計算結果のシェアを得ることができる．本研究での提案方式は，各グループごとに割り当てられた関数 f を計算する際に，各グループが $\mathcal{F}_{\mathsf{LSSSMPC}}^{f}$ を呼び出すことで関数 f の安全な計算を実現する．

理想機能： $\mathcal{F}_{\mathsf{LSSSMPC}}^{f}$

パラメータ：

- 計算対象の関数 f，秘密分散法の方式 LSSS，参加者集合 $\mathcal{P} \coloneqq \{P_1, P_2, \ldots, P_n\}$

入力：

- 各パーティ $P_i \in \mathcal{P}$ の保持する m 個のシェア $[x_1]_i, [x_2]_i, \ldots, [x_m]_i$

動作：

- $\mathcal{F}_{\mathsf{LSSSMPC}}^{f}$ は各パーティ $P_i \in \mathcal{P}$ からの入力

表 1 先行研究と提案方式の比較．n は入力者数，ℓ は暗号プリミティブ（FHE など）に依存するメッセージ長，c は committee に属するノード数，d はネットワーク構造を表す木の深さを表す．ここでは代表的なパラメータのみ記載し，他は省略している．NIMPC に関しては任意関数に対する一般構成の場合を記載．

Table 1 Comparison between prior works and the proposed scheme. n denotes the number of input parties, ℓ the message length depending on the cryptographic primitive (e.g., FHE), c the number of nodes in a committee, and d the depth of the tree representing the network structure. Only representative parameters are shown here, while others are omitted. For NIMPC, the case of the general construction for arbitrary functions is presented.

方式	プロトコル全体の通信量	入力者の通信量	全参加者常時オンライン	Fluidity	既存プロトコルの Black-box 利用	関数の制限
BGW ([2] 等)	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	要	non-maximal	△	なし
DN [7], ATLAS [9]	$\mathcal{O}(n)$	$\mathcal{O}(n)$	要	non-maximal	△	なし
Boyle et al. [4]	$\mathcal{O}(n \cdot \ell \cdot \mathrm{polylog}(n))$	$\mathrm{polylog}(n)$	要	non-maximal	×	なし
Fluid MPC [5]	$\mathcal{O}(c^2)$	$\mathcal{O}(c)$	不要	maximal	△	なし
YOSO [8]	$\mathcal{O}(n)$	$\mathcal{O}(c)$	不要	maximal	×	なし
NIMPC [1]	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	不要	maximal	×	なし
T-MPC [6]	$\mathcal{O}(ncd)$	$\mathcal{O}(c)$	不要	non-maximal	○	あり
提案方式	$\mathcal{O}\left(n \cdot n^{1/d}\right)$	$\mathcal{O}\left(n^{1/d}\right)$	不要	non-maximal	○	あり

$[x_1]_i, [x_2]_i, \ldots, [x_m]_i$ を待つ．

- honest なパーティは正しく送る
- corrupt されたパーティは攻撃者が設定した値を送る（もしくは送らないなどの不正な動作を行う）

- $\mathcal{F}^f_{\mathsf{LSSSMPC}}$ は各パーティから受け取った受け取ったシェアを用いて内部で秘密値 $x_1, x_2, \ldots, x_m$ を復元する．続いて $f(x_1, x_2, \ldots, x_n)$ を計算し，LSSS に従いシェア $[f(x_1, x_2, \ldots, x_m)]_i$ $(i \in [n])$ を生成し，各 P_i $(i \in \{1, 2, \ldots, n\})$ に送る．ただし，入力が正しく揃わず復元できない場合は $\perp$ を返す．

3.2 Linked Sharing

階層型秘密計算では，子ノード側のグループで得られた計算結果 y を，親ノード側のグループに安全に引き継ぐ仕組みが必要となる．提案方式では，子ノード側のグループと親ノード側のグループのそれぞれのグループで，同じ乱数 r を秘密分散して各ノードが r のシェアを保持するという構造を用いる．子ノード側のグループでは y を r でマスクした値 $y + r$ を親ノードに送信し，親ノードは自身の属するグループで $y + r$ を秘密分散し，各シェアから親ノード側で保持している r のシェアを各自が差し引くことで y のシェアを計算する．

このような同一の乱数のシェアを複数回使用するという発想は，BGW プロトコル [2] における再分散処理にも現れるものである．本稿では，同じ値に復元できるという関係により二つのグループを接続するという視点から，便宜的にこの構造を linked sharing と呼ぶことにする．

定義 1 (linked sharing)．体 $\mathbb{F}$ 上の値 x に対し，2 つの参加者集合 $\mathcal{P}, \mathcal{Q}$ において，それぞれに x を復元可能な秘密分散法のシェア $\{[x]_P\}_{P\in\mathcal{P}}, \{[x]_Q\}_{Q\in\mathcal{Q}}$ が与えられているとき，$\left(\{[x]_P\}_{P\in\mathcal{P}}, \{[x]_Q\}_{Q\in\mathcal{Q}}\right)$ を x の linked sharing と呼ぶ．また，x が $\mathbb{F}$ 上で一様ランダムに選ばれた値の場合を特に random linked sharing (RLS) と呼ぶこととする．ここで，$\mathcal{P}$ と $\mathcal{Q}$ のサイズは同じである必要はなく，また，しきい値などの復元条件も両者で異なっていてもよい．

RLS を生成する理想機能 $\mathcal{F}_{\mathsf{RLSGen}}$ とプロトコル Π_{RLSGen} をそれぞれ以下に示す：

理想機能：$\mathcal{F}_{\mathsf{RLSGen}}$

パラメータ：

- 体 $\mathbb{F}$，参加者集合 $\mathcal{P} = \left\{P_1, P_2, \ldots, P_{n_{|\mathcal{P}|}}\right\}$ および $\mathcal{Q} = \left\{Q_1, Q_2, \ldots, Q_{n_{|\mathcal{Q}|}}\right\}$，線形秘密分散法の方式 LSSS

動作：

- $\mathcal{F}_{\mathsf{RLSGen}}$ は $\mathbb{F}$ から一様ランダムに値 r を選ぶ
- $\mathcal{F}_{\mathsf{RLSGen}}$ は LSSS に従い，r のシェア $[r]^{(\mathcal{P})}_1, [r]^{(\mathcal{P})}_2, \ldots, [r]^{(\mathcal{P})}_{n_{|\mathcal{P}|}}$ を生成し，各 $P_i \in \mathcal{P}$ にシェア $[r]^{(\mathcal{P})}_i$ を送信する．
- $\mathcal{F}_{\mathsf{RLSGen}}$ は LSSS に従い，r のシェア $[R]^{(\mathcal{Q})}_1, [r]^{(\mathcal{Q})}_2, \ldots, [R]^{(\mathcal{Q})}_{n_{|\mathcal{Q}|}}$ を生成し，各 $Q_i \in \mathcal{Q}$ にシェア $[r]^{(\mathcal{Q})}_i$ を送信する．

プロトコル： Π_{RLSGen}
パラメータ：

- 体 $\mathbb{F}$，参加者集合 $\mathcal{P} = \{P_1, P_2, \ldots, P_{n_\mathcal{P}}\}$ および $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_{n_\mathcal{Q}}\}$，線形秘密分散法の方式 LSSS

動作：

- 各パーティ $P_i \in \mathcal{P}$ がそれぞれ以下の操作を行う：
  - P_i はそれぞれ独立に $\mathbb{F}$ から一様ランダムに r_i を選択
  - P_i は LSSS に従い，r_i のシェア $[r_i]_1^{(\mathcal{P})}, [r_i]_2^{(\mathcal{P})}, \ldots, [r_i]_{n_{|\mathcal{P}|}}^{(P)}$ を生成し，各 $P_j \in \mathcal{P}$ に対し $[r_i]_j^{(\mathcal{P})}$ を送信する（$j = i$ の場合は自身で保持しておく）．
  - P_i は LSSS に従い，r_i のシェア $[r_i]_1^{(\mathcal{Q})}, [r_i]_2^{(\mathcal{Q})}, \ldots, [r_i]_{n_{|\mathcal{Q}|}}^{(Q)}$ を生成し，各 $Q_j \in \mathcal{Q}$ に対し $[r_i]_j^{(\mathcal{Q})}$ を送信する
- 各 P_i は $[R]_i^{(\mathcal{P})} \coloneqq \sum_{j=1}^{n_{|\mathcal{P}|}} [r_j]_i^{(\mathcal{P})}$ を計算する．
- 各 Q_i は $[R]_i^{(\mathcal{Q})} \coloneqq \sum_{j=1}^{n_{|\mathcal{Q}|}} [r_j]_i^{(\mathcal{Q})}$ を計算する．

定理 1. Π_{RLSGen} は secure authenticated channel 上で $\mathcal{P}$ と $\mathcal{Q}$ のそれぞれで少なくとも 1 人の honest な参加者が存在するとき，任意の semi-honest な攻撃者の任意の結託に対し，$\mathcal{F}_{\mathsf{RLSGen}}$ を実現する．

3.3 Masked to Shares

提案方式では，あるグループの計算結果 y を次のグループに引き継ぐ際に，結果を乱数 r でマスクして親ノードに送信する．次のグループでは，親ノードはマスクされた値 $y + r$ のシェアを他ノードに分配し，各ノードは $y + r$ のシェアから r のシェアを差し引くことで y のシェアを得る．この処理を「masked to shares」として形式化するために，マスクされた値から秘密分散シェアへの変換を行う理想機能 $\mathcal{F}_{\mathsf{MtS}}$ の定義を行う．理想機能 $\mathcal{F}_{\mathsf{MtS}}$ とプロトコル Π_{MtS} はそれぞれ以下の通りである：

理想機能： $\mathcal{F}_{\mathsf{MtS}}$
パラメータ：

- 体 $\mathbb{F}$，参加者集合 $\mathcal{P} = \{P_1, P_2, \ldots, P_{n_\mathcal{P}}\}$，線形秘密分散法の方式 LSSS

入力

- P_i の保持する値 $y + r \in \mathbb{F}$
- 各 $P_j \in \mathcal{P}$ の持つ乱数 r のシェア $[r]_j$

動作：

- $\mathcal{F}_{\mathsf{MtS}}$ は P_i からの $y + r$ の入力を待つ
- $\mathcal{F}_{\mathsf{MtS}}$ は各パーティ $P_j \in \mathcal{P}$ からの $[r]_j$ の入力を待つ
- $\mathcal{F}_{\mathsf{MtS}}$ は受け取った r のシェアから r を復元し，$y = (y + r) - r$ を計算
- $\mathcal{F}_{\mathsf{MtS}}$ は LSSS にしたがい，y のシェア $[y]_1, [y]_2, \ldots, [y]_{n_\mathcal{P}}$ を生成
- $\mathcal{F}_{\mathsf{MtS}}$ は各パーティ $P_j \in \mathcal{P}$ に $[y]_j$ を送信

プロトコル： Π_{MtS}
パラメータ：

- 体 $\mathbb{F}$，参加者集合 $\mathcal{P} = \{P_1, P_2, \ldots, P_{n_\mathcal{P}}\}$，線形秘密分散法の方式 LSSS

入力：

- P_i の入力 $m = y + r$
- 各 $P_j \in \mathcal{P}$ の持つ乱数 r のシェア $[r]_j$

動作：

- P_i は LSSS にしたがい $[m]_i$ $(i \in [n_\mathcal{P}])$ を生成し，各 P_j $(j \in [\mathcal{P}])$ に $[m]_j$ を送信
- 各 P_j はローカルで $[m]_j - [r]_j$ を計算し，$[y]_j \coloneqq [m]_j - [r]_j$ として出力

定理 2. Π_{MtS} は secure authenticated channel 上で，少なくとも 1 人の honest な参加者が存在するとき，任意の semi-honest な攻撃者の任意の結託に対し，$\mathcal{F}_{\mathsf{MtS}}$ を実現する．

4. 二段階秘密計算プロトコル [12]

ここでは階層型秘密計算の最小ケースである $d = 2$ の場合を具体例で説明する．特に，$k = 3$ の場合を考える．このとき，入力を持つ参加者は $n = k^d = 9$ 人であり，これらは 3 人ずつのグループ $\mathcal{V}_{2,0}, \mathcal{V}_{2,1}, \mathcal{V}_{2,2}$ に分割されている．

深さ 2 の各グループの操作． グループ $\mathcal{V}_{2,0}$ では各参加者 $v_{2,0}, v_{2,1}, v_{2,2}$ がそれぞれ入力 x_0, x_1, x_2 を保持しており，これらに対して関数 $f_{2,0}$ を LSSS ベース MPC プロトコルにより計算する．各参加者は計算結果 $y_{2,0} \coloneqq f_{2,0}(x_0, x_1, x_2)$ のシェアを得る．

次に，この計算結果を親ノード $v_{1,0}$ に引き継ぐ必要がある．しかし，シェアをそのまま $v_{1,0}$ に送信すれば $v_{1,0}$ が $y_{2,0}$ を復元できてしまう．そこで，事前に分配されている乱数 $r_{2,0}$ のシェアを用い，各参加者は自身が得た $y_{2,1}$ のシェアをマスクして $[y_{2,0} + r_{2,0}]_i$ の形で親ノード $v_{1,0}$ に送信する．これにより，$v_{1,0}$ は $y_{2,0}$ 自体を知ることなく $y_{2,0} + r_{2,0}$ を受け取る．

同様に，$\mathcal{V}_{2,1}$ から $v_{1,1}$ に対して $y_{2,1}+r_{2,1}$ が，$\mathcal{V}_{2,2}$ から $v_{1,2}$ に対して $y_{2,2}+r_{2,2}$ がそれぞれ送られる．

深さ 1 のグループの操作　グループ $\mathcal{V}_{1,0}$ に属する $v_{1,j}$ $(j \in \{0,1,2\})$ はそれぞれ $y_{2,j}+r_{2,j}$ を保持している．ここから，$y_{2,0}, y_{2,1}, y_{2,2}$ を明かすことなく，それらを入力とした関数 $f_{1,0}$ の計算結果 $y_{1,0} \coloneqq f_{1,0}(y_{2,0}, y_{2,1}, y_{2,2})$ を計算しなければならない．

ここで，事前に $v_{1,j}$ $(j \in \{0,1,2\})$ に $r_{2,0}$ のシェアが分配されていると仮定する．このとき，$v_{1,0}$ が $y_{2,0}+r_{2,0}$ のシェアを生成・分配し，各ノードが $y_{2,0}+r_{2,0}$ のシェアから $r_{2,0}$ のシェアを差し引くことで，$y_{2,0}$ 自体を明かすことなく，$y_{2,0}$ のシェアを得ることができる．この操作を $y_{2,1}+r_{2,1}$ および $y_{2,2}+r_{2,2}$ に対しても実行することで，各ノードはそれぞれ $y_{2,0}, y_{2,1}, y_{2,2}$ のシェアを得られる．

このようにして得られた $y_{2,0}, y_{2,1}, y_{2,2}$ のシェアを入力として LSSS ベース MPC プロトコルを実行することで，各ノードは計算結果 $y_{1,0}$ のシェア $[y_{1,0}]_j$ を得ることができる．

上述の構成において，深さ 2 の各グループでマスクするために用いた乱数のシェアが，当該グループに分配され，かつ，その親ノードが属する深さ 1 のグループにも分配されていればよく，これは事前にセットアップとして random linked sharing 生成を行っていればよい．

深さ 1 のグループでの計算が終わると，$\mathcal{V}_{1,0}$ の各ノードは $y_{1,0}$ のシェアを保持している．このシェアも，深さ 2 の場合と同様に linked sharing によってマスクして親ノードに送信できる．この操作を繰り返すことで，より深い階層構造へ自然に拡張できる．

5. 階層型秘密計算

本稿で提案する階層型秘密計算（Hierarchical MPC）は，二段階秘密計算 [12] を任意のステージ数に拡張したモデルである．階層型秘密計算ではプロトコル実行に携わるノードを複数のグループに分割し，グループごとの計算結果を次のグループに引き渡し，その結果を入力として次のグループが計算を進めていく，という方式である．これらのグループ間の関係は木構造と対応させて表現する．

本節では，まず木構造との対応関係を定義し，続いてノードごとの役割を説明する．さらに，階層型秘密計算で満たすべき正当性および安全性の要件について整理する．なお，本稿では完全 k 分木の構造に限って議論を進めるが，他の木構造でも同様の議論が可能である．また，簡単のため，提案方式は (n,n) しきい値法に基づき構成するが，一般の (k,n) しきい値法に対しても同様の構成が可能である．

5.1 木構造との対応関係

深さ d の完全 k 分木を $T_{d,k}$ と表し，深さ ℓ $(0 \le \ell \le d)$ に位置する k^ℓ 個のノードを順に $v_{\ell,0}, v_{\ell,1}, \ldots, v_{\ell,k^\ell-1}$ とする．整数 α $(0 \le \alpha \le k^{\ell-1})$ に対し，ノードの集合 $\mathcal{V}_{\ell,\alpha} \coloneqq \{v_{\ell,k\cdot\alpha+i}\}_{i\in\{0,1,\ldots,k-1\}}$ を定義する．この集合に属するノードはいずれも $v_{\ell-1,\alpha}$ を親に持つ兄弟ノードである．階層型秘密計算における各グループとそのグループに属するノードを，それぞれ $\mathcal{V}_{\ell,\alpha}$ および $\mathcal{V}_{\ell,\alpha}$ に属する各ノード $v_{\ell,k\cdot\alpha+j}$ と一対一対応させることで，グループと $\mathcal{V}_{\ell,\alpha}$ を同一視する．これにより，階層型秘密計算における各グループは兄弟ノードとして表現でき，また，各グループ間の依存関係は木構造における親子関係で表現できる．また，$T_{d,k}$ における葉ノードのそれぞれが入力を持つ参加者であるとする．なお，階層型秘密計算では $T_{d,k}$ における根ノードに対応させるノードは存在しないものとする．

5.2 階層型秘密計算で計算対象とする関数

深さ $\ell \in \{1,2,\ldots,d\}$ の各グループ $\mathcal{V}_{\ell,\alpha}$ $(\alpha \in \{0,1,\ldots,k^\ell-1\})$ が計算する関数を $f_{\ell,\alpha} : \mathbb{F}^k \to \mathbb{F}$ とし，その出力を $y_{\ell,\alpha}$ とする．

まず，深さ d では，各ノード $v_{d,k\cdot\alpha+i}$ $(i \in \{0,1,\ldots,k^d-1\})$ の保持する秘密値 $x_{k\cdot\alpha+i}$ を入力として，各グループ $\mathcal{V}_{d,\alpha}$ ごとに関数 $f_{d,\alpha}$ を計算し，出力 $y_{d,\alpha}$ を親ノード $v_{d-1,\alpha}$ に安全に送信する．続いて，深さ $\ell = d-1, d-2, \ldots, 1$ の順に，各グループ $\mathcal{V}_{\ell,\alpha}$ は深さ $\ell+1$ のグループから受け取った k 個の値を入力として関数 $f_{\ell,\alpha}$ を計算し，その出力を親ノード $v_{\ell-1,\alpha}$ に送信する．最終的に，深さ 1 での出力 $y_{1,0}$ がプロトコル全体の出力となる．このとき，深さ $\ell = d$ のノードの入力値 x_i に対して $y_{d+1,i} \coloneqq x_i$, $(i \in \{0,1,\ldots,k^d-1\})$ と定義すると，$\ell \in \{1,2,\ldots,d\}$ の各グループ $\mathcal{V}_{\ell,\alpha}$ の出力は

$$y_{\ell,\alpha} \coloneqq f_{\ell,\alpha}(y_{\ell+1,k\cdot\alpha}, y_{\ell+1,k\cdot\alpha+1}, \ldots, y_{\ell+1,(k+1)\cdot\alpha-1})$$

と表される．したがって，階層型秘密計算全体では関数

$$g(x_0, x_1, \ldots, x_{k^d-1}) = y_{1,0}$$

を計算する．この形で計算できる合成関数には，積（AND），和（OR），平均値や分散などの統計量計算，分散環境における集約処理など，多くの実用的かつ有用な関数がある．

5.3 安全性と正当性の要件

階層型秘密計算プロトコルの構成においては，各グループ $\mathcal{V}_{\ell,\alpha}$ が親ノード $v_{\ell-1,\alpha}$ に送信する $y_{\ell,\alpha}$ について，$\ell = 1$ の場合を除き，$y_{\ell,\alpha}$ は $v_{\ell-1,\alpha}$ に対してさえも秘匿されなければならない．さらに同時に，$y_{1,0}$ が正しく計算されることも保証されなければならない．

6. 階層型秘密計算プロトコルの構成

本節では，前節で定義した階層型秘密計算のモデルに基づき，提案方式としての具体的なプロトコル構成を示す．また，効率性の解析と今後の課題についても議論する．

6.1 提案方式の構成

本節では，深さ d の完全 k 分木構造に基づく一般的な階層型秘密計算プロトコルの構成方法を与える．

プロトコル： $\Pi_{\mathsf{H\text{-}MPC}}$

パラメータ：

- 体 $\mathbb{F}$，深さ $d\ (\geq 2)$ の完全 k 分木 $T_{d,k}$，線形秘密分散法の方式 LSSS，$\mathcal{V}_{\ell,\alpha}\ (\ell \in \{1,2,\ldots,d\}, \alpha \in \{0,1,\ldots,k^\ell-1\})$ で計算される関数 $f_{\ell,\alpha}$

入力：

- 深さ d のノード $v_{d,i}\ (i \in \{0,1,2,\ldots,k^d-1\})$ の保持する秘密値 x_i

動作：

1. セットアップ：
   - $\mathcal{V}_{\ell,\alpha}$ と $\mathcal{V}_{\ell-1,\lfloor\alpha/k\rfloor}\ (\ell \in \{2,3,\ldots,d\}, \alpha \in \{0,1,\ldots,k^{\ell-1}-1\})$ について，$\mathcal{V}_{\ell-1,\lfloor\alpha/k\rfloor}$ を $\mathcal{Q}$，$\mathcal{V}_{\ell,\alpha}$ を $\mathcal{P}$ として $\mathcal{F}_{\mathsf{RLSGen}}$ を呼び出し，各ノード $v_{\ell-1,\lfloor\alpha/k\rfloor\cdot k+i}\ (i \in \{0,1,\ldots,k-1\})$ は $[r_{\ell,\alpha}]_i^{(\mathcal{V}_{\ell-1,\lfloor\alpha/k\rfloor})}$ を，各ノード $v_{\ell,k\cdot\alpha+i}\ (i \in \{0,1,\ldots,k-1\})$ は $[r_{\ell,\alpha}]_{k\cdot\alpha+i}^{(\mathcal{V}_{\ell,\alpha})}$ を受け取る
2. 深さ $\ell = d$ での処理：
   - 各ノード $v_{d,k\cdot\alpha+i}\ (i \in \{0,1,\ldots,k-1\}, \alpha \in \{0,1,\ldots,k^{d-1}-1\}))$ は LSSS に従い $x_{k\cdot\alpha+i}$ のシェア $[x_{k\cdot\alpha+i}]_{k\cdot\alpha+j}\ (j \in \{0,1,\ldots,k-1\})$ を生成し，$v_{d,k\cdot\alpha+j}$ に送信する
   - 各ノード $v_{d,k\cdot\alpha+i}$ は $([x_{k\cdot\alpha+j}]_{k\cdot\alpha+i})_{j\in\{0,1,\ldots,k-1\}}$ を入力として関数 $f_{d,\alpha} : \mathbb{F}^k \to \mathbb{F}$ を計算する LSSS ベース MPC プロトコルの理想機能 $\mathcal{F}_{\mathsf{LSSSMPC}}^{f_{d,\alpha}}$ を呼び出し，$y_{d,\alpha} \coloneqq f_{d,\alpha}(x_{k\cdot\alpha}, x_{k\cdot\alpha+1}, \ldots, x_{(k-1)\cdot\alpha})$ のシェア $[y_{d,\alpha}]_{k\cdot\alpha+i}$ を受け取る
   - 各ノード $v_{d,k\cdot\alpha+i}\ (i \in \{0,1,\ldots,k-1\})$ は $[y_{d,\alpha}]_{k\cdot\alpha+i}^{(\mathcal{V}_{d,\alpha})} + [r_{d,\alpha}]_{k\cdot\alpha+i}^{(\mathcal{V}_{d,\alpha})}$ を計算し，親ノード $v_{d-1,\alpha}$ に送信する
3. 深さ $\ell = d-1, d-2, \ldots, 2$ での処理:
   - 各グループ $\mathcal{V}_{\ell,\alpha}\ (\alpha \in \{0,1,\ldots,k^{\ell-1}-1\})$：
     - 各ノード $v_{\ell,k\cdot\alpha+i}\ (i \in \{0,1,\ldots,k-1\})$ は $[y_{\ell+1,k\cdot\alpha+i}]_{k\cdot\alpha+i}^{(\mathcal{V}_{\ell+1,k\cdot\alpha+i})} + [r_{\ell+1,k\cdot\alpha+i}]_{k\cdot\alpha+i}^{(\mathcal{V}_{\ell+1,k\cdot\alpha+i})}$ から $y_{\ell+1,k\cdot\alpha+i} + r_{\ell+1,k\cdot\alpha+i}$ を復元する．
     - 各ノード $v_{\ell,k\cdot\alpha+i}$ は復元した $y_{\ell+1,k\cdot\alpha+i} + r_{\ell+1,k\cdot\alpha+i}$ を，$\mathcal{V}_{\ell,\alpha}$ の各ノード $v_{\ell,k\cdot\alpha+j}\ (j \in \{0,1,\ldots,k-1\})$ は各自が持つ linked share $[r_{\ell+1,k\cdot\alpha+i}]_{k\cdot\alpha+j}^{(\mathcal{V}_{\ell,\alpha})}$ を入力として $\mathcal{F}_{\mathsf{MtS}}$ を呼び出し，各ノード $v_{\ell,k\cdot\alpha+j}$ は $[y_{\ell+1,k\cdot\alpha+i}]_{k\cdot\alpha+j}^{(\mathcal{V}_{\ell,\alpha})}$ を受け取る．
     - 各ノード $v_{\ell,k\cdot\alpha+i}$ は $([y_{\ell+1,k\cdot\alpha+j}]_{k\cdot\alpha+i})_{j\in\{0,1,\ldots,k-1\}}$ を入力として関数 $f_{\ell,\alpha} : \mathbb{F}^k \to \mathbb{F}$ を計算する LSSS ベース MPC プロトコルの理想機能 $\mathcal{F}_{\mathsf{LSSSMPC}}^{f_{\ell,\alpha}}$ を呼び出し，$y_{\ell,\alpha} \coloneqq f_{\ell,\alpha}(y_{\ell+1,k\cdot\alpha}, y_{\ell+1,k\cdot\alpha+1}, \ldots, y_{\ell+1,(k+1)\cdot\alpha-1})$ のシェア $[y_{\ell,\alpha}]_{k\cdot\alpha+i}$ を受け取る
     - 各ノード $v_{\ell,k\cdot\alpha+i}$ は $[y_{\ell,\alpha}]_{k\cdot\alpha+i}^{(\mathcal{V}_{\ell,\alpha})} + [r_{\ell,\alpha}]_{k\cdot\alpha+i}^{(\mathcal{V}_{\ell,\alpha})}$ を計算し，親ノード $v_{\ell-1,\lfloor\alpha/k\rfloor}$ に送信する
4. 深さ $\ell = 1$ での処理：
   - 各ノード $v_{1,i}\ (i \in \{0,1,\ldots,k-1\})$ は $[y_{2,i}]_i^{(\mathcal{V}_{2,i})} + [r_{2,i}]_i^{(\mathcal{V}_{2,i})}$ から $y_{2,i} + r_{2,i}$ を復元する．
   - 各ノード $v_{1,i}$ は復元した $y_{2,i} + r_{2,i}$ を，$\mathcal{V}_{1,0}$ の各ノード $v_{1,j}\ (j \in \{0,1,\ldots,k-1\})$ は各自が持つ linked share $[r_{2,i}]_j^{(\mathcal{V}_{1,0})}$ を入力として $\mathcal{F}_{\mathsf{MtS}}$ を呼び出し，各ノード $v_{1,j}$ は $[y_{2,i}]_j^{(\mathcal{V}_{1,0})}$ を受け取る．
   - 各ノード $v_{1,i}$ は $([y_{2,j}]_i^{(\mathcal{V}_{1,0})})_{j\in\{0,1,\ldots,k-1\}}$ を入力として関数 $f_{1,0} : \mathbb{F}^k \to \mathbb{F}$ を計算する LSSS ベース MPC プロトコルの理想機能 $\mathcal{F}_{\mathsf{LSSSMPC}}^{f_{1,0}}$ を呼び出し，$y_{1,0} \coloneqq f_{1,0}(y_{2,0}, y_{2,1}, \ldots, x_{y,k-1})$ のシェア $[y_{1,0}]_i$ を受け取る
   - 各ノード $v_{1,i}$ は $[y_{1,0}]_i$ を出力する

定理 3. $\Pi_{\mathsf{H\text{-}MPC}}$ は

$$\left(\begin{array}{l}\mathcal{F}_{\mathsf{RLSGen}},\ \mathcal{F}_{\mathsf{MtS}},\\ \left\{\mathcal{F}_{\mathsf{LSSSMPC}}^{f_{\ell,\alpha}}\right\}_{\ell\in\{1,\ldots,d\},\ \alpha\in\{0,\ldots,k^\ell-1\}}\end{array}\right)\text{-hybrid model}$$

において，secure authenticated channel 上で，semi-honest な攻撃者の任意の結託に対して安全である．ただし，各グループで採用される LSSS ベース MPC プロトコルの前提とする安全性要件（honest-majority など）を満たすことを前提とする．

Proof sketch. 各グループで採用される LSSS ベース MPC プロトコルの前提とする安全性要件が満たされているという前提から，RLS の秘密値（乱数）はいずれも結託者からは復元できず，各自のシェアも一様乱数と区別できない．また，$\mathcal{F}_{\mathsf{LSSSMPC}}^{f_{\ell,\alpha}}$ の出力のシェアも秘密値から独立な一様乱数と区別できない．よって，親ノードに送信する値と，それらから復元した値もそれぞれ一様乱数と区別できない．さらに，$\mathcal{F}_{\mathsf{MtS}}$ の呼び出しで得られるシェアも，いずれも一

様ランダムな値と区別がつかない．したがって，結託者の view は理想世界と識別できない． □

6.2 通信複雑性の解析

本節では，通信量を体 $\mathbb{F}$ 上の要素数で測り，グループ毎の MPC プロトコルの実行，RLS 生成，masked to shares プロトコルの実行，親ノードへの要素の送信，に分けてそれぞれ評価する．なお，提案方式は各グループ毎に異なる LSSS ベース MPC プロトコルを実行できる構成であるが，すべてのグループが同一の関数を計算し，さらに同一の MPC プロトコルを実行するものとして評価を行う．

$T_{d,k}$ に基づき，各ノードでの分岐数 k，深さを d とし，各グループのノード数を k とする．入力を持つノードの総数 n と k, d の間には $n = k^d$ が成り立つ．兄弟ノードの集合をグループとする定義より，深さ ℓ でのグループ数は $k^{\ell-1}$ であるため，総グループ数は $(k^d - 1)/(k - 1)$ となる．また，ステージを跨ぐ通信の回数は $(k^{d+1} - k^2)/(k - 1)$ であり，本稿で採用する linked sharing 生成プロトコル Π_{RLSGen} の実行回数は $(k^d - 1)/(k - 1)$ である．

これらの値を用いて，例えば各グループが BGW [2] のような参加者数 m に対して通信量が $\mathcal{O}\left(m^2\right)$ のプロトコルを実行する場合，Π_{RLSGen} の実行に関して $\mathcal{O}\left((2k-1)\cdot k\cdot(k^d-1)/(k-1)\right) = \mathcal{O}\left(k^{d+1}\right)$，入力値のシェア分配および Π_{MtS} の実行に関して $\mathcal{O}\left(k\cdot(k-1)\cdot(k^d-k)/(k-1) = \mathcal{O}\left(k^{d+1}\right)\right)$，各グループでの MPC プロトコルの実行に関して $\mathcal{O}\left(k^2\right)\cdot(k^d-1)/(k-1) = \mathcal{O}\left(k^{d+1}\right)$，子ノードから親ノードへの値の送信に関して $\mathcal{O}\left(1\cdot(k^{d+1}-k^2)/(k-1)\right) = \mathcal{O}\left(k^d\right)$ であることより，全体での通信複雑性は $\mathcal{O}\left(\max(k^{d+1}, k^{d+1}, k^{d+1}, k^d)\right) = \mathcal{O}\left(k^{d+1}\right) = \mathcal{O}\left(n\cdot n^{1/d}\right)$ となる．

また，入力を保持する各参加者の通信複雑性は，Π_{RLSGen} の実行に関して $\mathcal{O}(k)$，入力値のシェア分配に関して $\mathcal{O}(k)$，MPC プロトコルの実行に関して $\mathcal{O}(k)$，親ノードへの値の送信に関して $\mathcal{O}(1)$ であることより，$\mathcal{O}(\max(k, k, k, 1)) = \mathcal{O}(k) = \mathcal{O}\left(n^{1/d}\right)$ となる．同様に，深さ $\ell = d-1, d-2, \ldots, 1$ の各ノードの通信複雑性も $\mathcal{O}\left(n^{1/d}\right)$ と計算される．

6.3 今後の課題

対案方式は深さ d の完全 k 分木に基づき構成されているが，以下の拡張が考えられる．

第一に，分岐数を入力規模に対して対数程度まで小さく設計する方針である．この場合，木の深さは増加する一方，各参加者が通信するノードの数は対数程度に抑えられるため，各参加者の通信負荷はさらに小さくなる．第二に，深さごとに分岐数を可変にする方針である．例えば，葉に近い深さでは分岐を小さくして入力を持つ参加者の通信量を抑え，根に近い深さでは分岐を大きくすることで木の深さを抑えることで，各参加者の通信量と全体の通信量のバランスを調整できる．第三に，サーバ的を排除する拡張として，各グループから代表ノードをランダムに選出する設計が考えられる．この場合，選ばれた参加者のオンライン時間は増えるが，期待値としては緩やかな増加にとどまり，全参加者常時オンライン要件の解消に有効である．

謝辞 本研究は JSPS 科研費 JP23H00468, JP23H00479, JP23K17455, JP23K21644, JP23K24846, JP22H00521, JP25KJ1293, JST CREST JPMJCR22M1, JPMJCR23M2, AIP チャレンジプログラムの助成，および文部科学省の卓越研究員事業の支援を受けたものです．

参考文献

[1] Beimel, A., Gabizon, A., Ishai, Y., Kushilevitz, E., Meldgaard, S. and Paskin-Cherniavsky, A.: Non-interactive secure multiparty computation, *CRYPTO 2014*, Springer, pp. 387–404 (2014).

[2] Ben-Or, M., Goldwasser, S. and Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation, *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, p. 1–10 (1988).

[3] Blakley, G. R.: Safeguarding cryptographic keys, *Managing requirements knowledge, international workshop on*, IEEE Computer Society, pp. 313–313 (1979).

[4] Boyle, E., Goldwasser, S. and Tessaro, S.: Communication locality in secure multi-party computation: how to run sublinear algorithms in a distributed setting, *Theory of Cryptography Conference*, Springer, pp. 356–376 (2013).

[5] Choudhuri, A. R., Goel, A., Green, M., Jain, A. and Kaptchuk, G.: Fluid MPC: secure multiparty computation with dynamic participants, *CRYPTO 2021*, Springer, pp. 94–123 (2021).

[6] Clark, M. R. and Hopkinson, K. M.: Transferable multiparty computation with applications to the smart grid, *IEEE Transactions on Information Forensics and Security*, Vol. 9, No. 9, pp. 1356–1366 (2014).

[7] Damgård, I. and Nielsen, J. B.: Scalable and unconditionally secure multiparty computation, *Annual International Cryptology Conference*, Springer, pp. 572–590 (2007).

[8] Gentry, C., Halevi, S., Krawczyk, H., Magri, B., Nielsen, J. B., Rabin, T. and Yakoubov, S.: Yoso: You only speak once: Secure mpc with stateless ephemeral roles, *Annual International Cryptology Conference*, Springer, pp. 64–93 (2021).

[9] Goyal, V., Li, H., Ostrovsky, R., Polychroniadou, A. and Song, Y.: ATLAS: efficient and scalable MPC in the honest majority setting, *CRYPTO 2021*, Springer, pp. 244–274 (2021).

[10] Lindell, Y.: How to simulate it–a tutorial on the simulation proof technique, *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, pp. 277–346 (2017).

[11] Shamir, A.: How to share a secret, *Communications of the ACM*, Vol. 22, No. 11, pp. 612–613 (1979).

[12] 杉本航太，大原一真，渡邉洋平，岩本 貢：大規模なシステムに適した二段階秘密計算プロトコル，*SCIS 2025*, pp. 3C1–2 (2025).