事前計算が効率的で不正者が多くても安全なマルチパーティ計算

濱田 浩気　　菊池 亮

NTT セキュアプラットフォーム研究所
180-8585 東京都武蔵野市緑町 3-9-11
{hamada.koki,kikuchi.ryo}@lab.ntt.co.jp

あらまし 不正者が参加者の半数以上の場合にも安全なマルチパーティ計算において，事前計算を効率的に行う方法を提案する．近年マルチパーティ計算の研究が進み，特に不正者が参加者の半数を下回る場合に安全な手法については，実用的な速度の方法が提案されてきている．しかしながら，半数以上の参加者が結託してしまうと情報を復元できてしまうことから適用場面が限定され，例えば情報を自組織の外に出すことが許されない場合には用いることができなかった．これに対して，自身以外のすべての参加者が結託しても安全なマルチパーティ計算が提案されている．この手法は所望のマルチパーティ計算の実行前にあらかじめ事前計算をしておくことにより，不正者が参加者の半数を下回る場合に安全な手法と同程度の実行時の速度を達成している．その一方で，実行前の事前計算が膨大であることが実用上の問題となっている．本稿では，計算を手助けする情報を提供する信頼できるサーバの存在を仮定することで，事前計算を効率化する方法を提案する．さらに，複数のサーバのうち半数未満が不正者である場合にも適用できるように手法の拡張を行う．

Commodity-Based Secure Multi-party Computation

Koki Hamada　　Ryo Kikuchi

NTT Secure Platform Laboratories
3-9-11, Midori-cho Musashino-shi, Tokyo 180-8585 Japan
{hamada.koki,kikuchi.ryo}@lab.ntt.co.jp

Abstract We propose an MPC scheme that improves the effeciency of an existing MPC scheme for dishonest majority. Recently, many practical MPC schemes have been proposed. Although schemes that are secure against dishonest majority are desirable in terms of practicality, most of the schemes are secure only when majority of the parties are honest. A few MPC schemes for dishonest majority are proposed, but they require huge preprocessing computation time. In this paper, we address the problem by introducing trusted server that significantly reduces the running time of preprocessing. We also extend our scheme to the case when there are multiple servers.

1　はじめに

近年，個人に関する様々な情報を容易に取得できる環境が整い，データ分析技術の進歩と相まって，個人に関する情報の活用への期待が高まっている．その一方で，個人情報保護やプライバシの観点から個人に関する情報は極めて慎重な取扱いが必要とされ，データの保護と活用をどう両立させるかが問題となっている．

このようなデータの保護と活用の両立を目指して，プライバシ保護データ分析 (Privacy-Preserving Data Analysis: PPDA) 技術の研究が盛んになってきている．PPDA には，匿名

化技術と呼ばれる，データの有用性を保ったままプライバシー侵害が難しくなるようにデータを加工する技術や，データを処理する際に誰もデータを見ることなく処理結果を計算するマルチパーティ計算などがある．

1.1 マルチパーティ計算

マルチパーティ計算は秘密計算や秘匿関数計算，マルチパーティプロトコルとも呼ばれる，Yao による基本的なアイディア [10] を端緒とする技術であり，暗号化などの方法でデータを秘匿化したまま一度も元のデータに戻すことなく任意の計算を行う．マルチパーティ計算を用いることで，通常のデータ分析と同等の結果を高いプライバシ保護の下で得ることができる．しかし，マルチパーティ計算では通常の計算機上の計算に比べて処理速度が低下してしまうことが実用上の課題となっている．

処理速度の低下の大きな要因は基本演算のオーバーヘッドである．マルチパーティ計算ではデータの秘匿性を保つために，乗算のような通常の計算機では一命令で実行可能な基本演算にも複雑な処理を必要とし，その結果，処理時間も大きくなってしまう．これに対しては近年改良が進んでおり，Bogdanov らによる Sharemind[4]，Ben-David らによる FairplayMP[3]，Burkhart らによる SEPIA[5]，Geisler による VIFF[7]，Henecka らによる TASTY[8]，濱田らによる MEVAL [11] など，効率のよい基本演算を備えたフレームワークの提案，実装が行われている．

しかしながら，これまでの多くの手法では半数以上の参加者が結託してしまうと情報を復元できてしまうことから適用場面が限定され，例えば情報を自組織の外に出すことが許されない場合には用いることができなかった．

1.2 Damgård らの方式 [6]

これに対して，Damgård らにより，自身以外のすべての参加者が結託しても安全なマルチパーティ計算が提案されている [6]．この方式は秘密分散に基づいたマルチパーティ計算であり，能動的な攻撃者による n パーティのうち $n-1$ 人までの買収に対しても安全な方式である．

この方式は，Beaver の回路ランダム化 [1] と呼ばれる技法を用いて，MAC 付き 3 つ組と呼ばれる特殊な値を消費して乗算を効率良くかつ安全に実行する．MAC 付き 3 つ組は入力とは無関係であることからマルチパーティ計算が始まる前に事前計算により蓄積しておくことで，不正者が参加者の半数を下回ることを必要とする高速なマルチパーティ計算とも同程度の実行時の速度を達成している．

しかしながら，この MAC 付き 3 つ組の計算は somewhat 準同型暗号を使っていて非常に時間がかかることが問題となっている．MAC 付き 3 つ組の準備は実行時の処理に比べて約 1.09 万倍時間がかかり [6]，例えば体の大きさが 2^{64} の 2 パーティプロトコルの場合，10 秒間のマルチパーティ計算の実行のために約 30 時間の事前計算が必要となっていて，実用のためには事前計算の効率化が不可欠となっている．

1.3 本研究の貢献

本稿では，計算を手助けする情報を提供する信頼できるサーバの存在を仮定することで，Damgård らの方式 [6] で事前計算を効率化する方法を提案する．また，提案手法を拡張し，m 台のサーバのうち半数未満が不正者であっても安全とする方法の提案も行う．

また，提案手法の有効性の確認のため，手法を実装して実験によって性能の行った．その結果，1Gbps の LAN 環境とラップトップ PC で [6] に比べて約 1500 倍の事前計算の高速化を確認できた．

2 Damgård らの方式 [6] の概要

まず，提案手法が基づくマルチパーティ計算の方式である Damgård らの方式 [6] を説明する．本論文では，すべての計算は有限体 $\mathbb{F}_p$ 上で n 人のパーティ $P_1, \ldots, P_n$ で行われることとする．

[6] の方式は秘密分散された値を入力として，所望の計算結果の秘密分散された値を計算するマルチパーティ計算である．データの秘匿性は，加法的秘密分散により実現される．

攻撃者としては能動的な攻撃者を仮定しており，$n-1$ までのパーティを買収することのできる攻撃者に対しても安全である．

2.1 加法的秘密分散

[6] の方式は加法的秘密分散と呼ばれる方式に基づいたマルチパーティ計算である．加法的秘密分散は秘密分散共有法の一種で，秘密の値 $s \in \mathbb{F}_p$ を $s = s_1 + \cdots + s_n$ $(s_i \in \mathbb{F}_p)$ となるような n 個の値 $s_1, \ldots, s_n$ に分割し，各パーティ P_i が s_i を持つことで，すべてのパーティの持つ値 s_i をすべて集めない限り s を復元できないようにする方法である．

2.1.1 分散と復元

加法的秘密分散は分散と復元と呼ばれる 2 つの処理からなる．分散は秘密分散したい値 a を入力として，パーティ P_j が $a = a_1 + \cdots + a_n$ となる $a_1, \ldots, a_n \in \mathbb{F}_p$ をランダムに選び，各 a_i を P_i に送る処理である．このとき，値の組 $(a_1, \ldots, a_n)$ を $[\![a]\!]$ と表記し，a の秘匿文，a を $[\![a]\!]$ の平文と呼ぶ．また，a_i を $[\![a]\!]$ の P_i のシェアと呼び，$[\![a]\!]_i$ とも表記する．

秘匿文 $[\![a]\!]$ から，対応する平文である a を復元する演算の実行を

$$a \leftarrow \mathsf{Reveal}([\![a]\!])$$

と記述する．

2.1.2 秘匿文の線形計算

加法的秘密分散の秘匿文は線形性を有している．すなわち，秘匿文同士の加減算と定数倍が容易に実現できる．2 つの値 $a, b \in \mathbb{F}_p$ の秘匿文 $[\![a]\!]$ と $[\![b]\!]$ に対し，$[\![d]\!]_i := [\![a]\!]_i + [\![b]\!]_i$，$[\![e]\!]_i := [\![a]\!]_i - [\![b]\!]_i$ とすると，$[\![d]\!]$，$[\![e]\!]$ はそれぞれ $a+b$，$a-b$ の計算結果の秘匿文となっている．また，$a \in \mathbb{F}_p$ の秘匿文 $[\![a]\!]$ とすべてのパーティが知る定数 $c \in \mathbb{F}_p$ に対し，$[\![f]\!]_i := c[\![a]\!]_i$ とすると，$[\![f]\!]$ は ca の計算結果の秘匿文となっている．これらの演算の実行をそれぞれ，

$$[\![d]\!] \leftarrow \mathsf{Add}([\![a]\!], [\![b]\!]),$$
$$[\![e]\!] \leftarrow \mathsf{Sub}([\![a]\!], [\![b]\!]),$$
$$[\![f]\!] \leftarrow \mathsf{CMul}([\![a]\!], c)$$

と記述する．記述の簡潔化のため，誤解を招く恐れのない場合は，これらの演算の実行をそれぞれ，$[\![a]\!] + [\![b]\!]$，$[\![a]\!] - [\![b]\!]$，$c[\![a]\!]$ と略記する．

2.2 MAC

計算の正当性は，秘匿文 $[\![x]\!]$ に対して MAC と呼ぶ値の秘匿文 $[\![\gamma(x)]\!]$ を付与しておくことで実現する．各パーティ P_i は，MAC のキーと呼ぶランダムな固定の値 α の秘匿文のシェア $[\![\alpha]\!]_i$ を持っており，x の MAC$\gamma(x)$ は $\gamma(x) = x\alpha$ で定義される．攻撃者は MAC のキーを知らずに正当な MAC を維持した改ざんをすることが難しく，計算の正当性が担保される．

2.3 乗算

乗算は，Beaver による回路ランダム化 [1] を利用して行われる．具体的には，$[\![x]\!]$ と $[\![y]\!]$ の乗算を行う場合には，$ab = c$ を満たすランダムな秘匿文 $([\![a]\!], [\![b]\!], [\![c]\!])$ を用意し，$\epsilon \leftarrow \mathsf{Reveal}([\![x]\!] - [\![a]\!])$，$\rho \leftarrow \mathsf{Reveal}([\![y]\!] - [\![b]\!])$，で ϵ と ρ を計算し，これを使うと $c + \epsilon b + \rho a + \epsilon\rho = xy$ となることを利用し，$[\![c]\!] + \epsilon[\![b]\!] + \rho[\![a]\!] + \epsilon\rho$ を出力とする．MAC も同様に ϵ と ρ を使って計算することができる．この計算は 2 回 $\mathsf{Reveal}(\cdot)$ 以外はローカルでの計算のみで実現できるため，非常に効率がよい．

2.4 事前計算

ただし，上述の乗算を行うためには，1 回ごとに $ab = c$ を満たすランダムな秘匿文の組 $([\![a]\!], [\![b]\!], [\![c]\!])$，より正確には，MAC 付きの秘匿文の組 $(([\![a]\!], [\![\gamma(a)]\!]), ([\![b]\!], [\![\gamma(b)]\!]), ([\![c]\!], [\![\gamma(c)]\!]))$ (これを MAC 付き 3 つ組と呼ぶ) を 1 組消費する必要がある．そのため，乗算の実行前には MAC 付き 3 つ組を作る必要がある．

MAC 付き 3 つ組は乗算の入力とは無関係であるため，マルチパーティ計算を行う前に事前計算により大量に作成して保管しておくことができる．これにより，入力を受け取ってからの処理を非常に軽量に実現している．ただし，[6] では MAC 付き 3 つ組の作成には somewhat 準同型暗号を使っており，入力を受け取ってからの処理に比べて 1 万倍程度の時間を必要としている [6].

2.5 主要な部品

2.5.1 コミットメント

コミットメントは送信者であるパーティP_iが値vを他のパーティに秘密に提出するコミットと，コミットした値を後ですべてのパーティに明らかにするオープンの 2 つの処理からなる．送信者以外のパーティは値がオープンされるまでコミットされた値を知ることはできず，また，送信者がコミットした値は送信者も含めて誰も改ざんすることができない．

P_iが値vをコミットする処理を

$$\tau_v \leftarrow \mathsf{Commit}(v)$$

と記述する．τ_vはコミットした値vに紐づいたタグであり，全てのパーティに送られる．タグτ_vに紐づいたコミットされた値をP_iがオープンする処理を

$$\mathsf{Open}(\tau_v)$$

と記述する．この処理により，すべての参加者がタグτ_vに紐づいたコミットされた値vを得る．

本稿では，ランダムオラクルの存在を仮定して，[6] で使われている UC 安全なコミットメントを用いる．

2.5.2 DataCheck と MACCheck[6]

続けて使用される手続きで，$ab = c$を満たしていないかもしれない MAC 付きの秘匿文の組 $(([\![a]\!], [\![\gamma(a)]\!]), ([\![b]\!], [\![\gamma(b)]\!]), ([\![c]\!], [\![\gamma(c)]\!]))$ $2N$ 組を入力とし，N 組を消費して残りの N 組が$ab = c$を満たしているかどうかを確認する処理で，本稿でもアルゴリズムの一部として使用する．

計算コストは$\mathsf{Reveal}(\cdot)$が$2N$回，$\mathsf{Commit}(\cdot)$が$3n$回，$\mathsf{Open}(\cdot)$が$3n$回である．

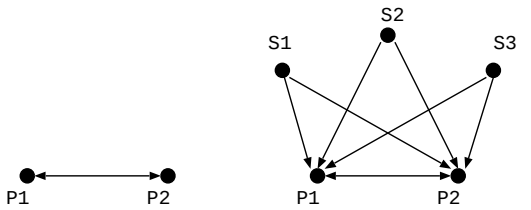
3 提案手法

Damgård らの手法 [6] の事前計算を効率化する手法を提案する．[6] では，MAC 付き 3 つ組と呼ばれるランダムな値を事前に大量に準備しておき，マルチパーティ計算の実行時にはこの 3 つ組を乗算 1 回ごとに 1 組消費して軽量のマルチパーティ計算を実現している．しかしながら，事前計算は somewhat 準同型暗号を用いていて非常に時間がかかる．そこで本稿では，信用できるサーバの存在を仮定し，その助けを借りて事前計算を効率よく実現する．

提案手法では，まず，サーバから MAC 付き 3 つ組を作るための部品となる MAC なし 3 つ組を大量に受け取る．そして，各パーティはこの MAC なし 3 つ組を用いて MAC 付き 3 つ組を効率よく生成する．

簡単のため，まずはサーバ数が$m = 1 (t = 0)$の場合を述べる．$m > 1$の場合への拡張方法は，3.4 節で議論する．

3.1 モデル



(a) 普通の MPC　(b) サーバが支援 (提案方式)

図 1: マルチパーティ計算 (MPC) の形態

提案手法では，Beaver によって提案された commodity-based cryptography [2] と同様のモデルをとる．通常のマルチパーティ計算 (MPC) は参加するパーティ間で相互に通信をしながら所望の計算を行う (図 1a)．これに対して，提案方式や commodity-based cryptography では，パーティの他にm台のサーバの存在を仮定し，サーバが支援を行うことでマルチパーティ計算を効率化する (図 1b).

一方で，新たな信用できるサーバを必要とすることから，通常のマルチパーティ計算に比べると適用可能な場面が限定されてしまう．しか

Algorithm 1 3 つ組を使った乗算

Notation: $[\![z]\!] \leftarrow \mathsf{Mul}([\![x]\!], [\![y]\!])$

Data: $[\![x]\!]$, $[\![y]\!]$

Result: $[\![z]\!]$. ただし $z = (x + \Delta_x)(y + \Delta_y)$ で，Δ_x，Δ_y は攻撃者が任意に決めた値.

1: サーバから 3 つ組 $([\![a]\!], [\![b]\!], [\![c]\!])$ を受け取る.
2: $\epsilon \leftarrow \mathsf{Reveal}([\![x]\!] - [\![a]\!])$,
$\rho \leftarrow \mathsf{Reveal}([\![y]\!] - [\![b]\!])$.
3: $[\![z]\!] \leftarrow [\![c]\!] + \epsilon[\![b]\!] + \rho[\![a]\!] + \epsilon\rho$.

しながら，このような信用できるサーバの存在はそれほど非現実的ではなく，例えば，国など社会的に信頼されている機関がこのサーバの役割を担えばよい．また，提案方式ではサーバはマルチパーティ計算には参加せずに，計算の助けとなる情報を一方的に送信するだけである (図 1b)．マルチパーティ計算を行うパーティとの独立性が高いため，実現性は高い.

3.2 安全性

提案手法は [6] と同様に，能動的な攻撃者を仮定する．攻撃者は $n-1$ 個までのパーティと，t 個 $(2t < m)$ までのサーバを買収することができる．提案するアルゴリズムは $1/|\mathbb{F}_p|$ 以下の確率を除いて攻撃者に新たな情報を与えず，また，アルゴリズムの規程しない出力への改ざんは $1/|\mathbb{F}_p|$ 以下の確率を除いて検知して異常終了する.

3.3 アルゴリズム

3.3.1 3 つ組を使った乗算

まず，サーバから受け取る 3 つ組を使った乗算を構築する．本節で構成する乗算は，サーバから受け取った 3 つ組 $([\![a]\!], [\![b]\!], [\![c]\!])$ を一組消費して，入力の 2 つの秘匿文 $[\![x]\!]$，$[\![y]\!]$ の積の秘匿文 $[\![z]\!]$ を計算しようとするが，出力は攻撃者の決める任意の値 Δ_x，Δ_y に対して $z = (x+\Delta_x)(y+\Delta_y)$ を満たす $[\![z]\!]$ となる．アルゴリズムを Algorithm 1 に示す.

安全性 Algorithm 1 は $[\![x]\!] - [\![a]\!]$ を復元した値 ϵ と $[\![y]\!] - [\![b]\!]$ を復元した値 ρ を使い，入力の秘匿文 $[\![z]\!]$ を計算する．攻撃者は ϵ と ρ を見るが，

Algorithm 2 MAC 付き 3 つ組の作成

1: ランダムな秘匿文 $[\![x]\!]$ と $[\![\alpha]\!]$ を作り，$[\![\beta]\!] \leftarrow \mathsf{Mul}([\![x]\!], [\![\alpha]\!])$ を計算し，$([\![x]\!], [\![\alpha]\!], [\![\beta]\!])$ を MAC のキーとする.
2: ランダムなシェア $[\![\boldsymbol{a}]\!]$，$[\![\boldsymbol{b}]\!]$ を作成.
3: $[\![c_i]\!] \leftarrow \mathsf{Mul}([\![a_i]\!], [\![b_i]\!])$.
4: $[\![\gamma(a_i)]\!] \leftarrow \mathsf{Mul}([\![a_i]\!], [\![\alpha]\!])$,
$[\![\gamma(b_i)]\!] \leftarrow \mathsf{Mul}([\![b_i]\!], [\![\alpha]\!])$,
$[\![\gamma(c_i)]\!] \leftarrow \mathsf{Mul}([\![c_i]\!], [\![\alpha]\!])$.
5: $\mathsf{Check}([\![\boldsymbol{a}]\!] \,||\, [\![\boldsymbol{b}]\!] \,||\, [\![\boldsymbol{c}]\!], [\![\gamma(\boldsymbol{a})]\!] \,||\, [\![\gamma(\boldsymbol{b})]\!] \,||\, [\![\gamma(\boldsymbol{c})]\!])$．ただし，$\boldsymbol{a} \,||\, \boldsymbol{b}$ は $\boldsymbol{a}$ と $\boldsymbol{b}$ の連結.
6: DataCheck[6] を実行.
7: MACCheck[6] を実行.

Algorithm 3 MAC の確認

Notation: $\mathsf{Check}([\![\boldsymbol{a}]\!], [\![\gamma(\boldsymbol{a})]\!])$

Data: $[\![\boldsymbol{a}]\!]$，$[\![\gamma(\boldsymbol{a})]\!]$，MAC キー $([\![x]\!], [\![\alpha]\!], [\![\beta]\!])$

1: $[\![z]\!] \leftarrow \mathsf{Mul}(\sum[\![a_i]\!] + [\![x]\!], [\![\alpha]\!]) - (\sum[\![\gamma(a_i)]\!] + [\![\beta]\!])$.
2: $[\![w]\!] \leftarrow \mathsf{MulRand}([\![z]\!])$
3: P_i: $\tau_i \leftarrow \mathsf{Commit}([\![w]\!]_i)$.
4: P_i: $\mathsf{Open}(\tau_i)$ により σ_i を得る.
5: もし $\sum \sigma_i \neq 0$ なら異常終了する.

これらはいずれも攻撃者の知らないランダムな値 a，b でマスクされているため，攻撃者は新たな情報を得ることはできない.

正当性 正直なパーティが受け取る値は ϵ と ρ だけであり，攻撃者のできる改ざんも ϵ，ρ に任意の値を加える改ざんに限られる．攻撃者が ϵ，ρ をそれぞれに Δ_x，Δ_y を加えた値に改ざんすると，出力の $[\![z]\!]$ は $z = (x + \Delta_x)(y + \Delta_y)$ を満たす値となる.

コスト この計算では，サーバから受け取る 3 つ組を 1 個消費し，プロトコル内の通信量は 2 回の $\mathsf{Reveal}(\cdot)$ で $4(n-1)$ である.

3.3.2 MAC 付き 3 つ組の作成

Algorithm 1 を利用して，MAC 付き 3 つ組を N 組作成する．まず最初に，ランダムな 2 つの値の秘匿文 $[\![x]\!]$，$[\![\alpha]\!]$ の積 $[\![\beta]\!]$ を計算し，$([\![x]\!], [\![\alpha]\!], [\![\beta]\!])$ を MAC のキーとする.

Algorithm 4 乱数倍

Notation: $[\![c]\!] \leftarrow \mathsf{MulRand}([\![a]\!])$

Data: $[\![a]\!]$

1: 乱数のシェア $[\![b]\!]$ を作成.
2: $\mathsf{Mul}([\![a]\!], [\![b]\!])$ を実行し，出力.

続いて，値の改ざんは許容して，Algorithm 1 を使って $c_i = a_i b_i$ を満たすランダムな値の秘匿文の組 $([\![a_i]\!], [\![b_i]\!], [\![c_i]\!])$ を $2N$ 組作成する．次に，これらに MAC を付与して，まず，$\mathsf{Check}(\cdot)$ によりすべての値に同じ MAC キーによる MAC が付与されていることの確認を行う．続いて，DataCheck と MACCheck [6] により MAC を付けた値の 3 つ組が正しい乗算の 3 つ組になっていることの確認を行う．アルゴリズムを Algorithm 2 に示す.

安全性 $\mathsf{Mul}(\cdot,\cdot)$，DataCheck，MACCheck は安全であるから，攻撃者が新たな情報を得るとすれば $\mathsf{Check}(\cdot)$ の $\mathsf{Open}(\tau_i)$ により得られる σ_i からである．しかしながら，$\sum \sigma_i = w$ は $\mathsf{MulRand}([\![z]\!])$ により攻撃者の決めた値 Δ_z と攻撃者の知らないランダムな値 y を使って $w = (z+\Delta_z)y$ と表されるので，$\Delta_z = -z$ かつ $y \neq 0$ となった場合，すなわち $1/|\mathbb{F}_p|$ 以下の確率で z の値を知ることしかできない.

正当性 Algorithm 3 はすべての要素の MAC が正しく計算されているかどうかを確認する．MAC キーは $[\![\beta]\!] \leftarrow \mathsf{Mul}([\![x]\!], [\![\alpha]\!])$ により計算された $([\![x]\!], [\![\alpha]\!], [\![\beta]\!])$ で定まり，攻撃者の決めた Δ_α，Δ_x によって $\beta = (\alpha + \Delta_\alpha)(x + \Delta_x)$ となっている場合は $\alpha + \Delta_\alpha$ が MAC キーとしての役割を果たす．以後は簡単のため，$\Delta_\alpha = 0$ とする．各秘匿文 $[\![a]\!]$ に $[\![\alpha]\!]$ を乗じて，これを $[\![\gamma(a)]\!]$ と表記し，$\gamma(a)$ を a の MAC と呼ぶ．$\gamma(a) = (a + \Delta_a)(r + \Delta_r)$ の形となっている場合は，$a + \Delta_a$ が a の正しい値として扱われるので，$\Delta_a = 0$，すなわち，$\gamma(a) = a(r + \Delta_r)$ とみなす.

Algorithm 3 は，入力の MAC $\gamma(a_i) = a_i(r + \Delta_i)$ のすべての $\Delta_i = 0$ であることを確認する．より正確には，すべての $\Delta_i = 0$ ならば攻撃者が適切に振る舞えば必ず正常に終了することができ，そうでないならば，$1/|\mathbb{F}_p|$ の確率を除いて異

Algorithm 5 サーバから受け取る 3 つ組の合成

Result: $([\![a]\!], [\![b]\!], [\![c]\!])$

1: P_i: $f_i(0) = [\![a]\!]_i$，$g_i(0) = [\![b]\!]_i$ となるランダムな t 次多項式 f_i，g_i を作り，$[\![f(j)]\!]_i := f_i(j)$，$[\![g(j)]\!]_i := g_i(j)$ とする．
2: $[\![h(j)]\!] \leftarrow \mathsf{Mul}([\![f(j)]\!], [\![g(j)]\!])$ をサーバ j から受け取った 3 つ組を使って計算.
3: $[\![c]\!] = [\![h(0)]\!] \leftarrow \sum_{j=1}^{2t+1} \lambda_j(0)[\![h(j)]\!]$.
ただし，$\lambda_j(x) = \prod_{1 \le i \le 2t+1, i \neq j} \frac{x-i}{j-i}$.

常終了する．ここで計算される z は攻撃者が定める値 ρ，Δ，Δ_i を使って $z = (\alpha + \Delta)(\sum a_i + x) - (\sum a_i(\alpha + \Delta_i) + \beta) + \rho$ と改ざんされ，$z = \sum a_i(\Delta - \Delta_i) + \Delta x + \rho$ となる．もしも $\Delta = 0$ かつすべての Δ_i が $\Delta_i = 0$ であれば，$z = \rho$ であり，攻撃者が $\rho = 0$ とすることで正常に終了できる.

「$\Delta = 0$ かつすべての Δ_i が $\Delta_i = 0$」ではないと仮定する．$\Delta \neq 0$ の場合は，$z = 0$ とするには $x = \Delta^{-1}(\sum a_i(\Delta - \Delta_i) + \rho)$ としなくてはならず，これは x を正しく推測しなくては $z = 0$ とできないことを意味し，この推測が成功する確率は $1/|\mathbb{F}_p|$ である．また，$\Delta = 0$ の場合は，ある $\Delta_j \neq 0$ が存在し，$\Delta_j = \Delta$ ならば $\Delta \neq 0$ なので上の議論より成功する確率は $1/|\mathbb{F}_p|$，$\Delta_j \neq \Delta$ ならば，$z = 0$ とするには $a_j = (\Delta - \Delta_j)^{-1}(\sum_{i \neq j} a_i(\Delta - \Delta_i) + \Delta x + \rho)$ としなくてはならず，やはり推測が必要で成功する確率は $1/|\mathbb{F}_p|$ である.

コスト $\mathsf{Mul}(\cdot,\cdot)$ が $8N+3$ 回，$\mathsf{Reveal}(\cdot)$ が $2N$ 回，$\mathsf{Commit}(\cdot)$ が $4n$ 回，$\mathsf{Open}(\cdot)$ が $4n$ 回である．よって，十分大きい N 組の MAC 付き 3 つ組を作る場合の 1 組あたりの通信量は $60n - 36$ である.

3.4 サーバが複数ある場合への拡張

これまではサーバの数を $m = 1$ とし，この唯一のサーバは信用できるものと仮定していた．ここでは，サーバの数を $m = 2t + 1$，$t \geq 1$ として，t 個以下のサーバが不正を行う場合にも Algorithm 1 で用いる 3 つ組を作成する方法を提案する.

基本的なアイディアは，Shamir の秘密分散法 [9] を用いて復元に必要なシェアの個数が $t+1$ である閾値秘密分散でランダムに作った 2 個の秘匿分をそれぞれ分割することである．分割された秘匿文ごとに異なるサーバから受け取った 3 つ組を使った乗算を行ない，最後に秘匿文のまま復元の線形計算を行ない，乗算結果の秘匿文を得る．アルゴリズムを Algorithm 5 に示す．

安全性　攻撃者は t 個の 3 つ組を知っているため t 個のシェアを見ることができるが，復元には $t+1$ 個のシェアが必要であることから，一切の情報を得ることができない．

正当性　攻撃者の改ざんによりできることは，出力の 3 つ組 $([\![a]\!], [\![b]\!], [\![c]\!])$ が攻撃者の定める任意の値 Δ_a，Δ_b を使って $c = a'b'$，$a' = a + \Delta_a$，$b' = b + \Delta_b$ を満たすランダムな a，b とすることである．

Algorithm 1 はこのような 3 つ組を使った場合にも 3.3.1 節 と同様の議論が成立するため，サーバ数が $m = 2t+1$ の場合にも Algorithm 2 は正当である．

コスト　この計算では各サーバから 1 個，合計 m 個の 3 つ組を消費して，1 個の 3 つ組を作る．プロトコル内の通信は乗算 m 回分であり，$4m(n-1) + 3mn$ である．

よって，このアルゴリズムを使うとき，十分大きい N 組の MAC 付き 3 つ組を作る場合の 1 組あたりの通信量は $56mn - 32m + 36(n-1)$ である．

4　性能評価

提案手法の有効性を確認するため，提案手法を実装して処理時間を測定した．

4.1　実験環境

4 台のラップトップ PC を 1 台のハブを介して 1Gbps の有線 LAN で互いに接続し，実験を行った．提案方式は n パーティによるマルチパーティ計算を m 台のサーバにより支援する形態であるため，本来であれば各パーティとサーバごとに 1 台の PC を割り当てるべきであるが，PC の台数が限られていたため，次のようにして実験を行った．4 台のうちの 3 台の PC はパーティとして使用し，各パーティに対して 1 台ずつ割り当てた．残りの 1 台の PC はサーバとして使用し，各サーバプログラムはこの PC 上の異なるプロセスとして実行した．このため，サーバ間の通信が LAN を介した通信と比べて非常に高速になるが，提案方式はサーバ間での通信は一切行わないため，実験への影響は少ない．

3 台のパーティ用に用いた PC は，CPU が Intel Core i7-2640M 2.80GHz，メモリが 8 GB，SSD が 128 GB である．サーバ用に用いた 1 台の PC は，CPU が Intel Core i7-2620M 2.70GHz，メモリが 4 GB，SSD が 160 GB である．すべての PC の OS は Linux (Ubuntu 12.04) であり，プログラミング言語は C++ を，コンパイラは g++ 4.6.3 を用いて提案手法を実装した．

4.2　実験結果

パーティ数 n を $n \in \{2, 3\}$ の範囲で，サーバ数 m を $m \in \{1, 3\}$ の範囲で，体の大きさ $|\mathbb{F}_p|$ を $|\mathbb{F}_p| \approx 2^{32}, 2^{64}$ の範囲で，一度に作成する MAC 付き 3 つ組の数 N を $N \in \{1, 10^3, 10^4, 10^5, 10^6\}$ の範囲で，それぞれ変えながら処理に要する時間の測定を行った．測定結果を表 1 に示す．各測定結果は，10 回 ($N = 10^6$ の場合のみ 2 回) の試行の平均値である．

先行研究 [6] では $n = 2$，$p \approx 2^{32}$ の場合に MAC 付き 3 つ組 1 組あたり 1.955×10^{-2} 秒を要していたのに対して，提案手法は $N = 10^5$ 組まとめて作った場合は 1 組あたりに要する時間が 1.276×10^{-5} 秒であり，実測値で約 1500 倍の高速化が実現できている．

また，このとき提案手法の 1 組あたりの通信量は $N = 10^5$ が十分に大きいとみなすと $56mn - 32m + 36(n-1)$ 個の 32 ビットの値の通信，すなわち $116 \times 32 = 3712$ ビットである．よって遅延やローカルの計算が無視できる理想的な状況では 1 組あたり 3.457×10^{-6} 秒と見積もられる．実測値と比較すると 4 倍程度であり，本実装により理論値に近い性能が達成できていることが確認できる．

表 1: N 組の MAC 付き 3 つ組作成に要した実行時間 (秒)

n	m	$p \approx$	$N = 1$	$N = 10^3$	$N = 10^4$	$N = 10^5$	$N = 10^6$
2	1	2^{32}	0.014	0.030	0.152	1.276	18.819
2	1	2^{64}	0.013	0.054	0.341	3.989	40.299
2	3	2^{32}	0.027	0.066	0.321	2.581	40.819
2	3	2^{64}	0.027	0.130	0.617	8.641	84.767
3	1	2^{32}	0.015	0.037	0.200	1.594	25.638
3	1	2^{64}	0.015	0.063	0.452	5.565	50.196
3	3	2^{32}	0.029	0.086	0.391	3.631	53.671
3	3	2^{64}	0.029	0.140	0.915	12.180	106.945

5 まとめ

事前計算が効率的で，不正者が参加者の半数以上の場合にも安全なマルチパーティ計算を提案した．提案手法は計算を手助けする情報を提供する信頼できるサーバの存在を仮定し，先行研究の事前計算を効率化した．さらに，複数のサーバのうち過半数が信頼できる場合にも適用できるように手法の拡張を行った．

提案手法の有効性の確認のため，提案手法を実装し，LAN 環境で処理性能の測定を行った．その結果，$p \approx 2^{32}$，$n = 2$，$m = 1$ の場合は先行研究と比べて実測値で約 1500 倍の処理速度が達成できており，提案手法の有効性が確認できた．

参考文献

[1] Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, Vol. 576 of *Lecture Notes in Computer Science*, pp. 420–432. Springer, 1991.

[2] Donald Beaver. Commodity-based cryptography (extended abstract). In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pp. 446–455. ACM, 1997.

[3] Assaf Ben-David, Noam Nisan, and Benny Pinkas. FairplayMP: a system for secure multi-party computation. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM Conference on Computer and Communications Security*, pp. 257–266. ACM, 2008.

[4] Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. In Sushil Jajodia and Javier López, editors, *ESORICS*, Vol. 5283 of *LNCS*, pp. 192–206. Springer, 2008.

[5] Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas A. Dimitropoulos. SEPIA: Privacy-preserving aggregation of multi-domain network events and statistics. In *USENIX Security Symposium*, pp. 223–240. USENIX Association, 2010.

[6] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings*, Vol. 8134 of *Lecture Notes in Computer Science*, pp. 1–18. Springer, 2013.

[7] Martin Geisler. *Cryptographic Protocols: Theory and Implementation*. PhD thesis, University of Aarhus, 2010.

[8] Wilko Henecka, Stefan Kögl, Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. TASTY: tool for automating secure two-party computations. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM Conference on Computer and Communications Security*, pp. 451–462. ACM, 2010.

[9] Adi Shamir. How to share a secret. *Commun. ACM*, Vol. 22, No. 11, pp. 612–613, 1979.

[10] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pp. 162–167, 1986.

[11] 濱田浩気, 五十嵐大, 菊池亮, 千田浩司, 諸橋玄武, 冨士仁, 高橋克巳. 実用的な速度で統計分析が可能な秘密計算システム MEVAL. In *CSS*, pp. 1–8, 2013.