情報処理学会論文誌 Vol.55 No.9 1971–1991 (Sep. 2014)

部分的に小さな法を用いたマルチパーティ計算のビット演算効率化

加藤 遼[1,a)] 西出 隆志[2,b)] 吉浦 裕[1,c)]

受付日 2013年11月29日, 採録日 2014年6月17日

概要：マルチパーティ計算（MPC）の 1 つの問題点は参加者間で膨大な通信を必要とする点である．MPC の通信量は，シェアのビット数，つまり秘密分散の法 p のビット数に比例することから，小さな法を用いることで通信量を低減する手法を提案する．大小比較や等号判定など多くのプロトコルがビットの状態の演算（ビット演算）を多用している点に着目し，これらのビット演算の共通パターンを見い出す．そして，プロトコルの入出力データの法 p およびそのビット数 ℓ_p に対して，このビット演算パターンを $p' > \max(T', \lceil\sqrt{\ell_p}\rceil + 1, n)$ となるような法 p' で実行し，元のプロトコルの機能と安全を維持しながら，通信量を削減する．ただし，T' はビット演算における秘密値の上限であり，n は MPC における参加者数である．この手法により，$(2, 3)$ 閾値法で法 p が 64 ビットの場合，大小比較，等号判定，区間判定の通信量を各々 80 パーセント，70 パーセント，80 パーセント程度削減できる．

キーワード：秘密分散法，マルチパーティ計算，法変換

Accelerating Bit-wise Calculation in Multi-party Computation by Partially Using Small Prime

Ryo Kato[1,a)] Takashi Nishide[2,b)] Hiroshi Yoshiura[1,c)]

Received: November 29, 2013, Accepted: June 17, 2014

Abstract: The problem with secure multi-party computation (MPC) is that the parties have to transfer a large amount of data to each other. The amount of data transferred is proportional to the number of bits of shares, i.e., the number of bits of the prime p that is used in secret sharing. A scheme is presented here that reduces the amount of data transferred by partially using a small prime. Common patterns of bit calculations are found in many MPC protocols such as comparison and equality testing. Bit-wise calculation of any of these patterns can maintain the correctness and security of the original protocol by using a prime p' that satisfies $p' > \max(T', \lceil\sqrt{\ell_p}\rceil + 1, n)$, where ℓ_p is the number of bits of the prime p used for the original protocol, T' is the upper bound of secret values in the bit-wise calculation, and n is the number of parties. For Shamir's $(2, 3)$ secret sharing with a prime of 64 bits, the amount of data transferred in comparison, equality-testing, and interval-testing protocols can be reduced by 80, 70, and 80% respectively.

Keywords: secure multi-party computation

1. はじめに

個人情報や機密情報を保護しながら活用することが重要になっており，情報を秘匿したままその関数値を計算する秘匿計算が期待されている．本論文では秘匿計算の一手法として，Shamir の (k, n) 閾値秘密分散法に基づくマルチパーティ計算（MPC）を取り上げる．Shamir の (k, n) 閾値秘密分散法は，Z_p の元である秘密情報 s を，分散情報

1 電気通信大学情報理工学研究科
Graduate School of Informatics and Engineering, The University of Electro-Communications, Chofu, Tokyo 182–8585, Japan
2 筑波大学システム情報系
Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, Ibaraki 305–8577, Japan
a) rio@kato.com
b) nishide@risk.tsukuba.ac.jp
c) yoshiura@hc.uec.ac.jp

（シェア）に分割し，n 人の参加者に配布する．k 個以上のシェアを集めないと元の秘密情報を復元できないという意味で，情報の秘匿が可能である．MPC では，秘密情報 a，b を分散した状態で，a のシェアと b のシェアから，a と b の和積の計算や大小比較を行うなどの様々なプロトコルが知られており，幅広い応用が検討されている．しかし，MPC には大量の通信を必要とするという問題点がある．積プロトコルでは，n 人の参加者が相互に通信するため $n(n-1)$ 回の通信が必要となる．等号判定や大小比較プロトコルの処理コストは積プロトコルに換算した通信量と並列化を考慮したラウンド数で評価される．たとえば，文献 [9] の等号判定プロトコルでは，32 ビットの秘密情報 a と b に対する等号判定の場合，2,592 回もの積プロトコルが必要になり，参加者が 5 人のときの通信量は 6,480 バイトにも及ぶ．

等号判定や大小比較，区間判定などの MPC の代表的なプロトコルは，秘密の情報をシェアの形で入力し，秘密の情報をシェアの形で出力する．これらのプロトコルでは，入力秘密情報から出力秘密情報が直接算出されるのではなく，数段階の処理を経る．これらの数段階の処理では，入力情報からの計算により，秘密情報がシェアの形で算出され，これらの秘密情報から最終的な出力情報が算出される．プロトコルにおいて算出される秘密情報のうち最終的な出力情報以外のものを，本論文では中間秘密情報と呼ぶことにする．

MPC の通信量は，シェアのビット数，つまり秘密分散の法 p のビット数 ℓ_p に比例する．また，ℓ_p はプロトコルの入出力となる秘密情報を正しく表現するために，秘密情報のサイズより大きく設定される．しかし，プロトコルの入出力が ℓ_p ビットであっても，プロトコルの中間秘密情報のサイズは ℓ_p より小さい場合がある．そこで，本論文ではサイズの小さな中間秘密情報の表現に ℓ_p より小さなビット数を用いる．つまり初期の秘密分散時に用いた素数 p より小さな素数 p' を用いて秘密分散をやり直すことで通信量を削減する手法を提案する．しかしながら，この手法の実現には以下の解決すべき問題が存在する．

- 中間秘密情報は秘匿されているので，一般には，そのサイズは未知であり，p' を定めることができない．
- 秘密分散時には法が公開される．そのため仮に p' を決めることができたとしても，その公開は中間秘密情報のサイズを公開することになり，秘密漏洩につながる．
- 素数 p による秘密分散を p' による秘密分散に変換する処理に通信量が必要となり，通信量の低減につながるとは限らない．

本研究では，等号判定や大小比較，区間判定などの多くの基本的なプロトコルはビットの状態の演算（ビット演算）を内部で多用している点に着目する．そしてこれらのビット演算を $p' > \max(T', \lceil\sqrt{\ell_p}\rceil + 1, n)$ となるような法 p' で実行する手法を提案する．ただし，T' はビット演算における中間秘密情報の値の上限であり，n は MPC における参加者数である．提案手法が上記の問題を解決し，元のプロトコルの機能と安全性を維持していることを示す．本論文の新規性は部分的に小さな法を用いる手法を，既存のプロトコルに適用する具体的な手続きを提案したことであり，通信量を評価することで，既存のプロトコルよりも通信量が削減できることを示した点について進歩性が認められると考える．本論文で取り上げる既存のプロトコルは，Joint Random Number Bitwise-Sharing，Least Significant Bit および，それらを用いる等号判定，大小比較，区間判定である．なお，本論文は著者らのシンポジウム論文 [13] を発展させたものである．本論文のプロトコルは攻撃者が semi-honest な状況を想定している．

2. 表記規則

- k：秘密の復元に必要なシェアの個数．
- n：シェアの分散数．
- $\log x$：底が 2 の対数であり，すなわち $\log_2 x$．
- p：ℓ_p ビットで表現される素数．なお，$\ell_p = \lfloor \log p \rfloor + 1$．
- $s.i$：i は添え字．$s.1$ と $s.2$ は区別される．
- s_B：$s_B \in \{0, 1\}$．
- s_i：s の i ビット目（$s_i \in \{0, 1\}$）．s_1 が最下位ビットである．なお，$(a + b \pmod p)_i$ と記した場合，$GF(p)$ 上で a と b を加算した結果の i ビット目を示す．
- $[s]_p$：s を素数 p で分散したシェア．
- $[a]_p + [b]_p$：秘密情報 $a, b \in \mathrm{Z}_p$ の和のシェア $[a + b \pmod p]_p$ を得るプロトコル．
- $[a]_p \times [b]_p$：秘密情報 $a, b \in \mathrm{Z}_p$ の積のシェア $[a \times b \pmod p]_p$ を得るプロトコル．

3. 先行研究

3.1 Shamir(k, n) 閾値秘密分散法 [11]

秘密情報 $s \in \mathrm{Z}_p$ を定数とし，$r.i \in \mathrm{Z}_p$ を乱数とする多項式

$$f(x) = s + (r.1)x + (r.2)x^2 + \cdots + (r.(k-1))x^{k-1} \pmod p \quad (1)$$

をつくる．n 人の参加者 P_d に，シェア $f(d)$ を配布する $(1 \le d \le n)$．シェア $f(d)$ を k 個以上集めると $k-1$ 次の多項式が一意に定めるため，秘密情報 s が求まる．k 個未満のシェアからは，s についてまったく情報を得ることができない．

3.2 マルチパーティ計算（MPC）

秘密情報を秘匿したまま，秘密分散法で分散されたシェアから，その関数値のシェアを求める手法である．和プロトコル，積プロトコル，乱数生成プロトコルが最も基礎的な MPC である．また，それらの基礎的な MPC を組み合

わせて構成された上位プロトコルとして，大小比較プロトコルや，等号判定プロトコルなどがある．

3.3 和プロトコル

$[a]_p$ および $[b]_p$ から，$[c]_p = [a+b]_p$ を求める．$[c]_p$ を得るには各参加者が独立に Z_p 上でシェア $[a]_p$ と $[b]_p$ を足せばよい．Z_p 上の差 $[a-b]_p$ についても同様にして求めることができる．和プロトコルでは，参加者間の通信が不要であるため処理コストは無視できる．

3.4 積プロトコル [3], [7]

$[a]_p$ および $[b]_p$ から，$[a \times b]_p$ を求める．その処理コストは，参加者間の通信量として見積もられる．1 回の積プロトコルを実行するためには n 人の参加者間で相互に通信するため，$n(n-1)$ 回の通信が必要となる．ただし，$[a]_p$ と公開情報 $e \in \mathrm{Z}_p$ の積 $[c]_p = [a \times e]_p$ を得る場合，通信は不要である．

3.5 乱数生成プロトコル Joint Random Number Sharing（JRNS）

乱数のシェア $[r]_p$ を出力する．下記のステップ 1 で，参加者 $_i$ は暗号計算を用いて乱数 $r.i$ を生成し，それを分散する（$1 \le i \le n$）．ステップ 2 で各参加者からのシェアを足し合わせて $[r]_p$ を求め，ステップ 3 で出力する．r は $\{0, \cdots, p-1\}$ において一様に分布する乱数となることが知られている [2]．通信量は 1，ラウンドは 1 である．なお，乱数生成に用いる暗号計算としては，ストリーム暗号の乱数生成部分などがある．

JRNS
1: Player$_i$ が乱数 $[r.i]_p$ を分散する $(1 \le i \le n)$
2: $[r]_p \leftarrow [r.1]_p + \cdots + [r.n]_p$
3: $[r]_p$ を出力

3.6 秘密情報公開プロトコル（Reveal）

Shamir(k, n) 閾値秘密分散法と素数 p を用いて分散された秘密情報 s を，n 人の参加者のうち k 人以上に公開する．秘密情報 s を知りたい m 人（$k \le m \le n$）の参加者の各々が，自分の持つシェアを自分以外の $m-1$ 人の参加者に送る．その結果，m 人の参加者は，各々 k 個以上のシェアを持つ．各人は，これらのシェアからラグランジュ補間を用いて，秘密情報 s を算出する．

提案方式の正確性に関わる点を，ここで述べておく．Reveal は，$GF(p)$ 上で秘密分散された秘密を，$GF(p)$ 上で正しく復元することができる．すなわち，素数 p の値の大小にかかわらず，秘密分散時と復元時の p が等しければ，秘密情報の正しい復元が可能である [11]．

3.7 通信量とラウンド数

MPC では等号判定，大小比較，区間判定などの上位レベルプロトコルが知られている．これらのプロトコルは主に和と積プロトコルで構成されるため，それらの処理コストは，必要な積プロトコルの回数で評価される．処理コストの評価には 2 つの評価指標がある．1 つは通信量であり，積プロトコルの回数で表される．もう 1 つは，ラウンド数と呼ばれる並列処理を考慮した積プロトコルの実行回数である．ラウンド数の評価では，できる限り並列処理を行う．たとえば，$[a]_p \times [b]_p \times [c]_p \times [d]_p$ という処理において，$[a]_p \times [b]_p$ と $[c]_p \times [d]_p$ を並列に実行し，それぞれの結果を掛け合わせることで，通信量は 3，ラウンド数は 2 となる．

3.8 Joint Random Bit Sharing（JRBS）

1 ビットの乱数のシェア $[r_B]_p$ を得る．通信量は 2 回，ラウンドは 2 である [6]．

JRBS
1: $[r]_p \leftarrow$ JRNS
2: $[a]_p \leftarrow [r]_p \times [r]_p$
3: $a \leftarrow \mathrm{Reveal}([a]_p)$
4: $r'.1, r'.2 \leftarrow \sqrt{a} \pmod p$
5: $r' \leftarrow \min(r'.1, r'.2)$
6: $[r_B]_p \leftarrow ([r]_p / r' + 1)/2 \pmod p$
7: $[r_B]_p$ を出力

JRNBS（3.5 節）を用いて乱数 $[r]_p$ を生成する．ステップ 2，3 では $[r]_p$ の二乗を計算し，その結果を a として公開する．ステップ 4 で，その平方根を取る．素数 p を法とすると，a の平方根は 2 つ存在する．ステップ 5 で，2 つの平方根 $r'.1$ と $r'.2$ のうち小さい方を選び，r' とする．$r' \in \{r, p-r\}$ である．ステップ 6 で r' 1 ビットの乱数 $[r_B]_p$ を計算する．

提案方式の安全性の説明に関わる点を，ここで述べておく．JRBS の出力する乱数 r_B は $\{0,1\}$ において一様である．また，JRBS の過程から r_B の値を推定することはできない．これらの点は，法 p の値にかかわらず成立する [6]．

3.9 Bitwise Less-Than（BLT）[6], [9]

Z_p の 2 つの要素 a と b を，ビットに分解されたシェア $[a_i]_p$ と $[b_i]_p (1 \le i \le \ell_p)$ の形で入力し，$[a < b]_p$ を出力する．すなわち，$a < b$ ならば $[1]_p$，さもなければ $[0]_p$ を出力する．本プロトコルの通信量は $19\ell_p$，ラウンド数は 8 である．実現方法の詳細は 5.2 節にて説明する．なお，$[a_i]_p$ と $[b_i]_p (1 \le i \le \ell_p)$ のうち一方が平文であってもよい．一方が平文の場合は，通信量は $17\ell_p$，ラウンド数は 7 である [6]．

3.10 Joint Random Number Bitwise-Sharing（JRNBS）

乱数 $r \in \mathrm{Z}_p$ のビット分解されたシェア $[r_i]_p (1 \le i \le \ell_p)$

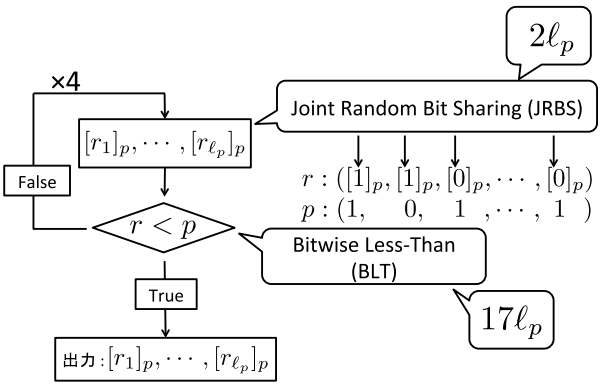

図 1 JRNBS プロトコル階層
Fig. 1 JRNBS protocol hierarchy.

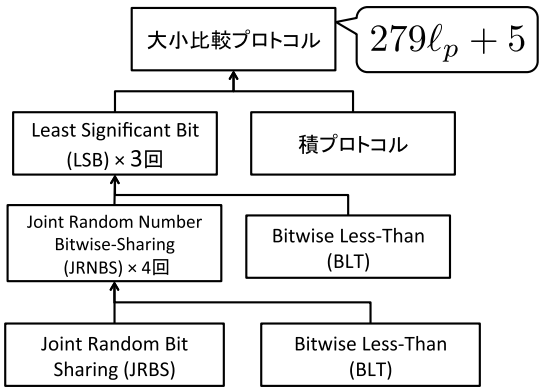

図 2 大小比較プロトコル階層
Fig. 2 Comparison protocol hierarchy.

を出力する．通信量は $76\ell_p$, 7 ラウンドである [9]．JRNBS の処理を下記に示す．また，その構造を**図 1** に示す．

JRNBS
1: $([r_1]_p, \cdots, [r_{\ell_p}]_p) \leftarrow$ JRBS を ℓ_p 回
2: $[v_B]_p \leftarrow \mathrm{BLT}(([r_1]_p, \cdots, [r_{\ell_p}]_p), (p_1, \cdots, p_{\ell_p}))$
3: $v_B \leftarrow \mathrm{Reveal}([v_B]_p)$
4: もしも $v_B = 1$（つまり，$r < p$）なら 5 へ進む．
さもなければ最初からやり直す．
5: $([r_1]_p, \cdots, [r_{\ell_p}]_p)$ を出力

本プロトコルは，ステップ 1 において，JRBS プロトコルを ℓ_p 回実行して，$[r]_p$ のビット表現 $([r_1]_p, \cdots, [r_{\ell_p}]_p)$ を生成する．ステップ 2 で，BLT プロトコルを用いて $[r]_p$ のビットと素数 p のビットを比較し，$r < p$ を判定する．なお，$(p_1, \cdots, p_{\ell_p})$ は p のビット表現である．ステップ 3 で判定結果の v_B を公開する．v_B は $r < p$ の真偽を表す 1 ビット情報である．$r < p$ が真であればステップ 5 で $([r_1]_p, \cdots, [r_{\ell_p}]_p)$ を出力，偽であれば，処理を最初からやり直す．このループの回数は生成される乱数 r と素数 p に依存するが，文献 [9] によるとコストの見積りにおいてはループの回数を 4 回と考えてよい．この 4 回のループは，並列実行が可能であり，ラウンド数 7 は並列実行を前提としている．全体の通信量（$76\ell_p$）のうち，BLT プロトコル 4 回分の通信量は $68\ell_p$ である．

3.11 大小比較プロトコル

$[a]_p$ および $[b]_p$ が与えられているとき，$[a > b]_p$ を出力する．すなわち，$a > b$ の真偽によって $[1]_p$ または $[0]_p$ を出力する．

大小比較プロトコルは様々な方式 [6], [9], [10] が提案されているが，通信量，ラウンド数ともに方式 [9] が最も小さいため，本研究ではこれを取り上げる．方式 [9] の通信量は $279\ell_p + 5$ であり，15 ラウンドである．

大小比較プロトコルは $a > \frac{p}{2}$ と $b > \frac{p}{2}$ と $(b - a \pmod p) > \frac{p}{2}$ を計算する．$a > \frac{p}{2}$ かつ $b < \frac{p}{2}$ のとき，$a > b$ である．また $a > \frac{p}{2}$ かつ $b > \frac{p}{2}$ かつ $(b - a \pmod p) > \frac{p}{2}$ のとき，$a > b$ である．詳細は省略するが，このように $a > b$ を別の条件判定の組合せに置き換えることで，$[a > b]_p$ を得る．**図 2** に示すように，LSB プロトコルを用いて，$[a > p/2]_p$ と $[b > p/2]_p$ と $[(b - a) > p/2]_p$ を計算し，その結果から，積プロトコルを用いて，$[a < b]_p$ を計算する．

以下では，LSB プロトコルによる $[a > p/2]_p$ と $[b > p/2]_p$ と $[(b - a) > p/2]_p$ の計算について，$[a > p/2]_p$ を例として説明する．LSB は，与えられたシェアの最下位ビットを求めるプロトコルである．$2a \pmod p$ の最下位ビットが 1 のとき，$2a > p$ すなわち $a > p/2$ であり，$2a \pmod p$ の最下位ビットが 0 とき，$a < p/2$ である．したがって，LSB を用いて，$[2a]_p$ の最下位ビットを求めることで，$[a > p/2]_p$ を求めることができる．

LSB プロトコルの流れを以下に示す．また，以下では $c = 2a$ とし，c の最下位ビットを求めるものとする．

LSB プロトコル
Input: $[c]_p$
1: $([r_1]_p, \cdots, [r_{\ell_p}]_p) \leftarrow$ JRNBS
2: $[r]_p \leftarrow ([r_1]_p, \cdots, [r_{\ell_p}]_p)$
3: $[v]_p \leftarrow [c]_p + [r]_p$
4: $v \leftarrow \mathrm{Reveal}([v]_p)$
5: $[u_B]_p \leftarrow \mathrm{BLT}((v_1, \cdots, v_{\ell_p}), ([r_1]_p, \cdots, [r_{\ell_p}]_p))$
6: $[d_B]_p \leftarrow v_1 \oplus [r_1]_p$
7: $[t_B]_p \leftarrow [u_B]_p \times (1 - [d_B]_p) + (1 - [u_B]_p) \times [d_B]_p$
8: $[t_B]_p$ を出力

LSB プロトコルは，上記のステップ 1，2 において，JRNBS プロトコルを用いて乱数 $[r]_p$ およびそのビット表現 $[r_i]_p (1 \le i \le \ell_p)$ を参加者間で共有する．次にステップ 3 および 4 において，$c + r \pmod p$ を v として公開する．ステップ 5 において，BLT プロトコルを用いて，$c + r < r$（すなわち $c + r > p$）の判定を行い，結果を $[u_B]p$ とする．u_B は $c + r > p$ の真偽を表す 1 ビットの情報である．$[u_B]_p = 0$ すなわち $c + r < p$ のとき，$v_1 = c_1 \oplus r_1$ であり，v_1 と $[r_1]_p$ が参加者間で共有されているため，$[c_1]_p = v_1[r_1]_p$ を計算することができる（ステップ 6）．なお，ステップ

6 の d_B は，$[u_B]_p = 0$ のときの c_1 を表す 1 ビット情報である．$[u_B]p = 1$ すなわち $c + r > p$ のときは，$(c + r \pmod p))_1 = \neg(c_1 \oplus r_1) = (1 - d_B)$ となるので，ステップ 7 により，$[u_B]_p = 1$ および 0 の両方の場合について，$[c_1]_p$ を算出する．

上記のステップ 1 の JRNBS のなかで BLT プロトコルを 4 回用いる．また，ステップ 5 でも BLT を用いるので，$[a > p/2]_p$ の計算に BLT を合計 5 回用いる．$[b > p/2]_p$ と $[(b-a) > p/2]_p$ の計算も同様であるため，大小比較プロトコル中で BLT プロトコルを 15 回用いることになり，BLT プロトコルの通信量は，$255\ell_p = 15 \times 17\ell_p$ となる．

3.12 等号判定プロトコル

$[a]_p$ および $[b]_p$ が与えられているとき，$[a = b]_p$ を得る．すなわち，$a = b$ の真偽によって $[1]_p$ または $[0]_p$ を出力する．

文献 [5] では，フェルマーの小定理を利用して等号判定プロトコルが可能であることを示している．この手法は文献 [4] で実装されている．しかし，この手法はラウンド数が ℓ_p に比例する．文献 [6] ではビット分解プロトコルを用いて $98\ell_p + 94\ell_p \log \ell_p$ 回，39 ラウンドの等号判定プロトコルが提案された．この手法はラウンド数が ℓ_p に比例しない．さらに文献 [9] では，処理コストが 81ℓ 回，8 ラウンドの手法が提案された．

本研究ではこれらのプロトコルのうちで，定数ラウンドかつ最も通信量が小さい文献 [9] の等号判定プロトコルを用いる．

等号判定プロトコル

Input: $[a]_p$, $[b]_p$
1: $([r_1]_p, \cdots, [r_{\ell_p}]_p) \leftarrow$ JRNBS
2: $[r]_p \leftarrow ([r_1]_p, \cdots, [r_{\ell_p}]_p)$
3: $[v]_p \leftarrow [a]_p - [b]_p + [r]_p$
4: $v \leftarrow \mathrm{Reveal}([v]_p)$
5: **for** $i := 1 \cdots \ell_p$ **do**
6: **if** $v_i = 0$ **then**
7: $[c_i]_p \leftarrow 1 - [r_i]_p$
8: **else**
9: $[c_i]_p \leftarrow [r_i]_p$
10: **end if**
11: **end for**
12: $[t_B]_p \leftarrow \mathrm{UnboundedFanInAnd}(([c_1]_p, \cdots, [c_{\ell_p}]_p))$
13: $[t_B]_p$ を出力

等号判定プロトコルは，ステップ 1，2 において，JRNBS プロトコルを用い，参加者間で乱数 $[r]_p$ およびそのビット表現 $[r_i]_p (1 \le i \le \ell_p)$ を共有する．ステップ 3，4 において，$[a-b]_p$ に乱数を加算して，$v = (a - b + r \pmod p)$ を公開する．$v = r$ と $a = b$ は等価である．したがって，ステップ 5 から 12 において，$v = r$ すなわち $v_i = r_i (1 \le i \le \ell_p)$ の判定を行う．詳細は省略するが，$[c_i]p = [v_i = r_i]p$ であり，Unbounded Fan-In And を用いて，$[c_1]p \cdots [c_{\ell_p}]p$ の論理積を計算することで，$[a = b]_p$ を求めることができる．なお，$[t_B]p$ は $[a = b]_p$ を表す 1 ビットのシェアである．等号判定プロトコルでは，JRNBS プロトコルが用いられるので，BLT プロトコルが 4 回用いられる．その通信量は $68\ell_p = 4 \times 17\ell_p$ である．

3.13 区間判定プロトコル

$[a]_p$ および公開値 b, c が与えられているとき，$[b < a < c]_p$ を得る．すなわち，$b < a < c$ の真偽によって $[1]_p$ または $[0]_p$ を出力する．

区間判定プロトコルは様々な方式 [6], [9] が提案されているが，通信量，ラウンド数ともに方式 [9] が最も小さいため，本研究ではこれを取り上げる．方式 [9] の通信量は $110\ell_p + 1$ であり，13 ラウンドである．このプロトコルの詳細を以下に記す．

区間判定プロトコル

Input: $[a]_p$, b, c
1: $([r_1]_p, \cdots, [r_{\ell_p}]_p) \leftarrow$ JRNBS
2: $[r]_p \leftarrow ([r_1]_p, \cdots, [r_{\ell_p}]_p)$
3: $[v]_p \leftarrow [a]_p + [r]_p$
4: $v \leftarrow \mathrm{Reveal}([v]_p)$
5: **if** $c \le v$ **then**
6: $d.1 = v - b$
7: $d.2 = v - c$
8: **end if**
9: **if** $v \le b$ **then**
10: $d.1 = v + p - c$
11: $d.2 = v + p - b$
12: **end if**
13: **if** $b < v < c$ **then**
14: $d.1 = v - b - 1$
15: $d.2 = v + p - c - 1$
16: **end if**
17: $[u_B.1]_p \leftarrow \mathrm{BLT}(([r_1]_p, \cdots, [r_{\ell_p}]_p), ((d.1)_1, \cdots, (d.1)_{\ell_p}))$
18: $[u_B.2]_p \leftarrow \mathrm{BLT}(([r_1]_p, \cdots, [r_{\ell_p}]_p), ((d.2)_1, \cdots, (d.2)_{\ell_p}))$
19: **if** $v \le b$ or $c \le v$ **then**
20: $[u_B]_p \leftarrow [u_B.1]_p \times [u_B.2]_p$
21: **end if**
22: **if** $b < v < c$ **then**
23: $[u_B]_p \leftarrow 1 - [u_B.1]_p \times [u_B.2]_p$
24: **end if**
25: $[u_B]_p$ を出力

本プロトコルは，上記のステップ 1 から 4 に示すように，JRNBS プロトコルを用いて乱数 $[r]_p$ およびそのビット表現 $[r_i]_p (1 \le i \le \ell_p)$ を参加者間で共有し，$[a]_p$ に乱数を加算して，$v = a + r$ を公開する．$c \le v$ と $b < v < c$ と $v \le b$ のそれぞれについて異なる処理を行う．ここでは，$c \le v$ の場合について説明する．このとき，詳細は省略するが，$b < a < c$ と $(a + r - c \pmod p) < r < (a + r - b \pmod p))$ は等価である．$v = (a + r \pmod p))$ と b と c が公開されているので，$(a + r - c \pmod p))$ と $(a + r - b \pmod p))$ を計算

することができる．$[r_i]_p$ は共有されているので，BLT プロトコルを 2 回用いて，$(a+r-c \pmod p) < r < (a+r-b \pmod p)$ の条件判定ができる．

$b < v < c$ や $v \leq b$ についても同様に，$b < a < c$ と等価な条件判定が存在する．ステップ 17 から 23 は，これら 3 つの条件判定をまとめて行っている．なお，u_B は，$b < a < c$ の真偽を表す 1 ビット情報である．区間判定では，JRNBS プロトコルに加え，BLT プロトコルが 2 回用いられているため，BLT プロトコルが合計 6 回用いられる．その通信量は $102\ell_p = 6 \times 17\ell_p$ である．

3.14 通信量の問題

MPC は参加者間の膨大な通信を要する．たとえば $\ell_p = 32$，参加者数が 5 の場合，1 回の等号判定の通信量は 6,480 バイトに及ぶ．また，3.11 節や 3.13 節で述べたように，多くの上位プロトコルは等号判定よりさらに多くの通信量を必要とする．そのため，MPC の通信量の削減は重要である．

4. 提案方式

4.1 基本方針と課題

1 章で述べたように，MPC の代表的なプロトコルは，秘密の情報をシェアの形で入力し，秘密の情報をシェアの形で出力する．ここで，入力となる秘密情報を入力秘密情報，出力となる秘密情報を出力秘密情報と呼ぶことにする．また，ある情報 A から別の情報 B を算出する場合に，情報 A を参照して情報 B を算出すると呼ぶことにする．一般に MPC のプロトコルでは，入力秘密情報から出力秘密情報が直接算出される場合は少なく，数段階の処理を経る．この処理には，以下の 4 種類がある．

(1) 入力秘密情報を参照して，別の秘密情報または平文情報を算出する．
(2) 入力秘密情報とは無関係に秘密情報を算出する．
(3) 秘密情報を公開する．すなわち参加者間で秘密情報のシェアを交換する．
(4) 入力秘密情報，上記の (1) と (2) で算出された秘密情報および平文情報，(3) で公開された平文情報のすべてあるいは一部を参照して，別の秘密情報または平文情報を算出する．

以上の処理で算出される秘密情報のうち出力秘密情報以外のものを，中間秘密情報と呼ぶことにする．中間秘密情報は，出力秘密情報を算出するために，プロトコルのある処理で算出され，以降の処理で参照される秘密情報である．なお，MPC のプロトコルの中には，入力秘密情報を持たず，出力秘密情報だけを持つものもある．3 章で述べたプロトコルのうち大小比較，等号判定，区間判定は入出力の両者を持ち，JRBS，JRNBS は出力のみを持つ．入力秘密情報を持たないプロトコルの場合は，上記の処理のうち (1) を含まない．

$[c]_p \leftarrow [a]_p \times [b]_p$ と $[c]_q \leftarrow [a]_q \times [b]_q$ は両者とも積プロトコル 1 回であるが，通信量は異なる．なぜなら，素数 p で分散されたシェアのサイズは ℓ_p ビットであり，素数 q で分散されたシェアのサイズは ℓ_q ビットであるためである．積プロトコルでは参加者間で，シェアを相互に通信するため，通信量はシェアのサイズに比例する．

この点に着目し，プロトコルの中間秘密情報の分散に，できるだけ小さな素数 p' を用いることで，通信量を削減する手法を考える．

しかし，このアプローチの実践には以下の課題を解決する必要がある．

課題 1：正確性の維持

中間秘密情報を s' とする．もし $s' \geq p'$ となると，$GF(p')$ 上の演算により s' の値が変わってしまい，改良前のプロトコルの機能を維持することができない．つまり正確性が失われる．そのため，$p' > s'$ が必要条件であり，その条件下で最小の素数であることが望ましい．しかし，s' は一般には秘匿された値である（各参加者は自分のシェア $[s']_p$ だけを知っている）ため，正確性を維持可能な p' の値を確定することができない．したがって，正確性を維持しながら p' の値を確定できる場合とその確定の方法を見い出す必要がある．

課題 2：安全性の維持

p から p' への法変換時には，p' の値が公開される．そのため，課題 1 を解決できたとしても，p' の公開により，中間秘密情報 s' が p' より小さいことが公開される．したがって，p' の公開が安全性の低下にならない方法を見い出す必要がある．

課題 3：全体としての通信量の削減

p から p' へ法変換を行うための処理が必要であり，そこで通信量が発生する．そのため，この通信量の増加を上回る通信量の削減効果が必要である．

4.2 既存プロトコルにおける通信量の分析

乱数生成，大小比較，等号判定などの重要な既存プロトコルを分析した結果，その通信量の大部分はビット演算であることが分かった．たとえば，3.11 節で述べた大小比較プロトコルの場合，積プロトコルと JRBS プロトコルを除いたすべての演算が，ビット演算（BLT: Bitwise Less-Than）である．これは，大小比較プロトコル全体の通信量の 90 パーセントにのぼる．また等号判定プロトコルについても同様で，JRBS プロトコル以外のプロトコルはすべてビット演算（BLT およびビットごとの等号判定）である．等号判定プロトコルの通信量におけるビット演算の占める割合は 90 パーセントである．

4.3 通信量削減の基本方針

ビットのシェアは1ビットの秘密情報である．しかし，既存プロトコルでは，プロトコルの入出力をなす多ビットの秘密情報と同じ素数 p で分散されている．そこで，既存プロトコル中のビット演算の部分で，p よりも小さい素数 p' を用いて中間秘密情報を分散する．必要に応じて，ビット演算の前または後（あるいは両方）で法変換処理を実行する．小さい法 p' の利用による通信量の削減が法変換処理の通信量よりも大きい場合に全体として通信量を削減することができる．

なお，大小比較プロトコルのように，出力秘密情報が1ビットであるプロトコルも存在する．この出力秘密情報を小さい法 p' で分散しても，出力値（$a > b$ か否か）は影響を受けない．しかし，入力秘密情報と出力秘密情報をともに法 p で分散していたプロトコルが，入力秘密情報は p，出力秘密情報は p' で分散するようになり，プロトコルの入出力の関係が変化してしまう．本論文では，従来のプロトコルの入出力を維持しながら通信量の削減を目指すので，出力秘密情報が1ビットの場合でも，これを p' で分散することはせず，中間秘密情報だけを p' で分散することとする．

4.4 正確性と安全性に関する考察

上記の課題1で述べたように，ビット演算部分で算出および参照される中間秘密情報 s' の値が p' 以上であると，$GF(p')$ 上の演算の影響で s' の値が変わってしまい，正確性が失われる．そこで s' の値は p' 未満である必要がある．しかし，乱数加算あるいは積算によって秘密の値を秘匿して公開する場合には，加算あるいは積算後の値と p' の大小関係は自由である必要がある．

たとえば，$[v]_{p'} = [s']_{p'} + [r]_{p'}$ によって s' に乱数 r を加算したのち，v を公開する場合，v は $GF(p')$ 上で一様な乱数にならなければならない．そのような v を作るためには，$(s' + r \pmod p)$ と p' の大小関係が固定されていてはならない．

また，3.8節で述べたように，Reveal プロトコルは，素数 p の値の大小にかかわらず，秘密分散時と復元時の p が等しければ，秘密情報を正しく復元することができる．これを小さい法 p' にあてはめると，Reveal は，p' の値にかかわらず，正しく動作する．したがって，p' の決定において Reveal を考慮する必要はない．以上から，正確性を維持するために，小さい法 p' を下記の手順で決める．

(1) p' を用いるビット演算部分のうち，乱数の和積によって値を秘匿する部分および Reveal 部分以外に着目する．当該着目部分における中間秘密情報の上限を T とする．T の値は，具体的なビット演算に依存して決まる．たとえば，5章に示すように，ビット演算が BLT（BitwiseLess-Than）の場合には，$T = \lceil \sqrt{\ell_p} \rceil + 1$ となる．また，A.1節に示すように ℓ_p 個のビットのシェアに対し Unbounded Fan-In And を用いる場合は，$T = \ell_p + 1$ となる．

(2) $\max(T, x_1, \cdots, x_n)$ を超える最小の素数を p' とする．ただし，$x_i (1 \le i \le n)$ は，秘密情報の分散に用いる多項式 (1) の入力値であり，n 人の参加者それぞれが異なる x_i を持つ．x_i の値によって安全性が低下することがないため，$x_i = i$ としてよい（$1 \le i \le n$）．このとき，p' は $\max(T, n)$ を超える最小の素数である．

上記のように p' を設定すれば，G の内部で中間秘密情報 s'_j が p' を超えることはないので，正確性を維持することができる．p' は入力となる秘密情報 s および中間秘密情報 s' に依存せずに，プロトコルの構造だけに依存して設定され，プロトコルの構造は公開されているので，p' によって s あるいは s' に関する情報が漏洩することはない．したがって，p の代わりに p' を用いることで安全性への影響はなく，既存プロトコルの安全性を維持することができる．

4.5 着目するビット演算

多くのプロトコルに共通して含まれ，かつ，通信量の大きいビット演算を取り上げ，その部分に小さい法を適用すれば効果的であると考えられる．そのようなビット演算として，3.10節で述べた JRNBS があげられる．3.11節〜3.13節で述べたように，JRNBS は，大小判定，等号判定，区間判定など多くの重要なプロトコルに共通して含まれ，かつ，通信量は $76\ell_p$ と大きい．

JRNBS

1: $([r_1]_p, \cdots, [r_{\ell_p}]_p) \leftarrow$ JRBS を ℓ_p 回
2: $[v_B]_p \leftarrow \mathrm{BLT}(([r_1]_p, \cdots, [r_{\ell_p}]_p), (p_1, \cdots, p_{\ell_p}))$
3: $v_B \leftarrow \mathrm{Reveal}([v_B]_p)$
4: もしも $v_B = 1$（つまり，$r < p$）なら5へ進む．
　さもなければ最初からやり直す．
5: $([r_1]_p, \cdots, [r_{\ell_p}]_p)$ を出力

また，7章で述べるように，大小判定，等号判定，区間判定では JRNBS の使い方が共通している．すなわち，これらの3つのプロトコルは下記の処理を共通に含んでいる．

JRNBS 利用パターン

Input: $[s]_p$
1: $([r_1]_p, \cdots, [r_{\ell_p}]_p) \leftarrow$ JRNBS
2: $[r]_p \leftarrow ([r_1]_p, \cdots, [r_{\ell_p}]_p)$
3: $[r]_p$ を用いて s を秘匿した値 $[v]_p$ を計算．
4: $v \leftarrow \mathrm{Reveal}([v]_p)$
5: $[u_B]_p \leftarrow \mathrm{F}((v_1, \cdots, v_{\ell_p}), ([r_1]_p, \cdots, [r_{\ell_p}]_p))$
6: $[u_B]_p$ を出力

上記の処理のうちステップ1は JRNBS を用いて，乱数 $([r_1]_p, \cdots, [r_{\ell_p}]_p)$ を生成する．ステップ2は，ビット単位のシェア $[r_i]_p$ から2進数の乱数のシェア $[r]_p$ を計算する．ステップ3は，乱数 $[r]_p$ を用いて秘密情報 s を秘匿した値

$[v]_p$ を計算し，ステップ 4 は値 v を公開する．$[v]_p$ の計算方法は，大小判定，等号判定，区間判定によって異なる．たとえば，大小比較の場合，$[v]_p \leftarrow [s]_p + [r]_p$ である．ステップ 5 の F は，ビット単位の平文とビット単位のシェアを各々 ℓ_p 個入力し，ビット値のシェアを出力する処理である．すなわち，ステップ 4 で開示された値 v のビット $(v_1, \cdots, v_{\ell_p})$ とステップ 1 で算出した $([r_1]_p, \cdots, [r_{\ell_p}]_p)$ を入力し，1 ビット値のシェア $[u_B]_p$ を出力する．このとき，$[u_B]_p$ の値（復元したときの u_B の値）は，乱数 $([r_1]_p, \cdots, [r_{\ell_p}]_p)$ の値にかかわらず，入力秘密情報 $[s]_p$ の値だけに依存して決まる．たとえば 3.11 節の大小比較プロトコルの中の LSB プロトコルにおいて，ステップ 5 から 7 が F に相当し，ステップ 7 の $[c_1]_p$ が F の出力 $[u_B]_p$ に相当する．ここで，$[c_1]_p$ は入力秘密情報 $[c]_p$ の最下位ビットのシェアであり，乱数の値にかかわらず，入力秘密情報 $[c]_p$ だけに依存して決まる．F の具体的な内容は，大小判定，等号判定，区間判定によって異なる．たとえば，大小比較の場合，F の具体的な内容は，Bitwise Less-than，ビット間の排他的論理和，ビット間の積から構成される処理である．上記の処理のうち，ステップ 3 における $[v]_p$ の算出方法とステップ 5 における F の内容以外は，大小判定，等号判定，区間判定に共通である．また，F の具体的な内容は異なるが，入力秘密情報 $[s]_p$ の値だけに依存して出力 $[u_B]_p$ が決まる点も共通している．そこで，上記を JRNBS を利用する共通のパターンであると考え，JRNBS 利用パターンと呼ぶことにする．

4.6 ビット法変換プロトコル

Shamir(k, n) 閾値秘密分散によるビット情報のシェアに対して法変換を行うマルチパーティプロトコルのうち汎用的なものとしては，文献 [1] に示される Conversion Between Bit Shares（CBBS）が知られている．CBBS では $p > n$ かつ $p' > n$ のとき，法 p と p' の間で，法変換が可能であり，1 ビットの秘密情報 $[s_B]_p$ を入力して，1 ビットの秘密情報 $[s_B]_{p'}$ を出力する．$[s_B]_p$ と $[s_B]_{p'}$ は同じ秘密の値を，異なる法 p と p' によって秘密分散したシェアである．

CBBS プロトコル

Input: $[s_B]_p$

1: Player$_i$ は乱数 $r_B.i$ を生成し，$[r_B.i]_p$ および $[r_B.i]_{p'}$ を分散する $(1 \le i \le k)$．
2: $[r_B]_p \leftarrow [r_B.1]_p \oplus \cdots \oplus [r_B.k]_p$
3: $[r_B]_{p'} \leftarrow [r_B.1]_{p'} \oplus \cdots \oplus [r_B.k]_{p'}$
4: $[t_B]_p \leftarrow [s_B]_p \oplus [r_B]_p$
5: $t_B \leftarrow \mathrm{Reveal}([t_B]_p)$
6: $[s_B]_{p'} \leftarrow [r_B]_{p'} \oplus t_B$
7: $[s_B]_{p'}$ を出力

文献 [11] に示される乱数生成の方式を用いて，ステップ 1 から 3 を計算し，1 ビットの乱数 r_B を共有する．ステップ 4，5 では，$[s_B]_p$ と $[r_B]_p$ の排他的論理和の結果 t_B（1 ビットの中間秘密情報）を公開する．ステップ 6 において，t_B と $[r_B]_{p'}$ の排他的論理和求から $[s_B]_{p'}$ を求める．通信量は $GF(p)$ において $5k+2$，$GF(p')$ において $5k+1$ である．ラウンド数は 4 である．

本プロトコルの安全性は，文献 [1], [11] に示されているが，本論文の提案手法の安全性に深く関係するため，以下で説明する．ステップ 5 において公開される t_B から，秘密情報 s_B に関する情報が漏洩しなければ，CBBS の安全性は保証される．t_B は乱数 r_B と s_B の排他的論理和の結果である．したがって，r_B が $\{0,1\}$ において一様な分布を持つ 1 ビット乱数であれば，s_B に関する情報は漏洩しない．乱数 r_B は各参加者（$1 \le i \le k$）が生成する乱数 $r_B.i$ の排他的論理和の結果である．一般に MPC は k 人以上の不正者が存在しないことを前提としているため，k 個の 1 ビット乱数 $(r_B.1, \cdots, r_B.k)$ のうちの 1 個以上は $\{0,1\}$ において一様な分布を持つ 1 ビット乱数である．したがってその排他的論理和である r_B は $\{0,1\}$ において一様な分布を持つ 1 ビット乱数である．

4.7 通信量の新しい指標

一般的に MPC では，あるプロトコルの通信量は，そのプロトコルの内部で積プロトコルが何回用いられたかで，見積もられる．たとえば，前述の大小比較プロトコルの通信量は $279\ell_p + 5$ だが，これは大小比較プロトコルの内部で，積プロトコルが $279\ell_p + 5$ 回相当用いられていることを意味していた．しかし，前述したとおり，積プロトコルの回数が同じでも，素数のサイズ（すなわち参加者間で通信するシェアのビット数）に比例して，通信量は大きくなる．そこで素数のサイズの影響を明示するために，従来の通信量の指標を拡張する．

本指標では，プロトコルの通信量を，プロトコル内部で用いられる積プロトコルの回数と秘密分散時の素数のビット数との積で見積もる．すなわち，通信量 = 積プロトコルの回数 × 素数のビット数，とする．たとえば，秘密分散時の素数を p とすると，積プロトコル $([c]_p \leftarrow [a]_p \times [b]_p)$ は，参加者間で ℓ_p ビットのシェアを通信する．そのため，従来の指標による積プロトコル 1 回の通信量は 1 であるが，本指標では $1 \times \ell_p$ となる．同様に秘密分散時の素数を q とすると，積プロトコル $([c]_q \leftarrow [a]_q \times [b]_q)$ の通信量は，本指標では $1 \times \ell_q$ となる．本指標を用いて，改めて大小比較プロトコルの通信量を見積もる．大小比較プロトコルでは，秘密分散時の素数を p とすると，積プロトコルが $279\ell_p + 5$ 回相当用いられる．また，個々の積プロトコルの通信量は ℓ_p である．そのため，大小比較プロトコルの通信量は $(279\ell_p + 5)\ell_p$ となる．同様に等号判定プロトコルの通信量は $81\ell_p^2$ となる．上述した CBBS の通信量は，$(5k+2)\ell_p + (5k+1)\ell_{p'}$ となる．

5. JRNBS の効率化

5.1 AJRNBS

小さい法 p' を用いて，JRNBS を効率化する．まず，4.5 節で示した JRNBS プロトコルを下記の Modified-JRNBS（MJRNBS）に等価変形する．

Modified-JRNBS（MJRNBS）

1: $([r_1]_p, \cdots, [r_{\ell_p}]_p) \leftarrow$ JRBS を ℓ_p 回
2: $[v_B]_p \leftarrow \mathrm{BLT}(([r_1]_p, \cdots, [r_{\ell_p}]_p), (p_1, \cdots, p_{\ell_p}))$
3: $v_B \leftarrow \mathrm{Reveal}([v_B]_p)$
4: もしも $v_B = 1$（つまり，$r < p$）なら 5 へ進む．
さもなければ最初からやり直す．
5: $([r_1]_p, \cdots, [r_{\ell_p}]_p)$ を $([u_1]_p, \cdots, [u_{\ell_p}]_p)$ にリネーム
6: $([u_1]_p, \cdots, [u_{\ell_p}]_p)$ を出力

MJRNBS を JRNBS と比べると，ステップ 5 において，秘密情報の名前 $r_i (1 \le i \le \ell_p)$ を u_i に変更し，ステップ 6 において，u_i および u を出力しているだけである．したがって，MJRNBS と JRNBS は，同じ値の範囲で同じ一様性を有する乱数を同じ個数出力する．また，両者の安全性も等しい．

次に，MJRNBS に小さい法 p' を取り入れ，Advanced-JRNBS（AJRNBS）を構成する．MJRNBS には入力秘密情報はなく，出力秘密情報はステップ 5，6 における $([u_1]_p, \cdots, [u_{\ell_p}]_p)$ である．中間秘密情報は，ステップ 1 から 3 におけるすべての秘密情報である．そこで，ステップ 1 から 3 における中間秘密情報を小さい法 p' によって秘密分散する．AJRNBS の通信量は，$(5k+1)\ell_p^2 + (5k+78)\ell_p\ell_{p'}$，ラウンド数は 8 である．以下に，AJRNBS を示す．

Advanced JRNBS（AJRNBS）プロトコル

1: $([r_1]_{p'}, \cdots, [r_{\ell_p}]_{p'}) \leftarrow$ JRBS を ℓ_p 回　▷ $2\ell_p\ell_{p'}$
2: $[v_B]_{p'} \leftarrow \mathrm{BLT}(([r_1]_{p'}, \cdots, [r_{\ell_p}]_{p'}), (p_1, \cdots, p_{\ell_p}))$　▷ $17\ell_p\ell_{p'}$
3: $v_B \leftarrow \mathrm{Reveal}([v_B]_{p'})$
4: もしも $v_B = 1$（つまり，$r < p$）なら 5 へ進む．
さもなければ最初からやり直す．
5: $[r_i]_p \leftarrow \mathrm{CBBS}([r_i]_{p'})$ を ℓ_p 回　▷ $(5k+1)\ell_p^2 + (5k+2)\ell_p\ell_{p'}$
6: $([r_1]_p, \cdots, [r_{\ell_p}]_p)$ を $([u_1]_p, \cdots, [u_{\ell_p}]_p)$ にリネーム
$([r_1]_{p'}, \cdots, [r_{\ell_p}]_{p'})$ を $([u_1]_{p'}, \cdots, [u_{\ell_p}]_{p'})$ にリネーム
7: $([u_1]_p, \cdots, [u_{\ell_p}]_p), ([u_1]_{p'}, \cdots, [u_{\ell_p}]_{p'})$ を出力

5.2 法 p' の決定

AJRNBS において法 p' を用いるビット演算は，ステップ 1（JRBS），2（BLT）3（Reveal）である．これらのステップに着目し，p' の値を決定する．

JRBS

3.8 節を $GF(p')$ 上で実行する場合を考える．JRBS では，乱数 $[r]_{p'}$ を生成し，r^2 を公開する．r^2 から $\sqrt{r^2} \pmod{p'}$ を計算し，$[(\frac{r}{\sqrt{r^2}} + 1)/2]_{p'}$ を求める．

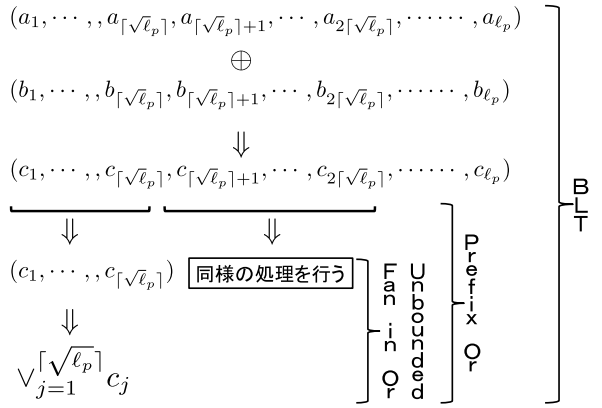

図 3 BLT 概要
Fig. 3 Outline of BLT.

JRNBS では r^2 が p を超えてしまう可能性が存在する．ただし，これは r の値を秘匿するための演算であり，p' の決定には影響しない．

BLT

BLT は $[a_i]_{p'}$ および $[b_i]_{p'} (1 \le i \le \ell_p)$ が与えられているとき，$[a > b]_{p'}$ を得る．$c_i = a_i \oplus b_i$ とすると，$(c_j, \cdots, c_{\ell_p}) = (1, 0, \cdots, 0)$ となるような最上位の 1 のビット $c_j (1 \le j \le \ell_p)$ を Prefix-Or プロトコルを用いて計算する．a_j が 1 なら $a > b$ であり，b_j が 1 なら $a < b$ である．p' の値は Prefix-Or プロトコルに依存する．

Prefix-Or

Prefix-Or は $[c_i]_{p'} (1 \le i \le \ell_p)$ が与えられているとき，$[\vee_{j=i}^{\ell_p} c_j]_{p'} (1 \le i \le \ell_p)$ を得る．なお，$\vee$ は 1 ビット情報 c_j を真理値と見なしたときの論理和を表す．その方法として，$[c_i]_{p'}$ を $\lceil\sqrt{\ell_p}\rceil$ 個ごとのブロックに区切って，それぞれのブロックに対して Unbounded Fan-In Or を用いることで，再計算を避けつつ，定数ラウンドを実現する．p' の値は Unbounded Fan-In Or に依存する．後述する Unbounded Fan-In Or では，$[c_i]_{p'} (1 \le i \le \lceil\sqrt{\ell_p}\rceil)$ のブロックを対象にして説明する．

Unbounded Fan-In Or [9]

Unbounded Fan-In Or は $[c_i]_{p'} (1 \le i \le \lceil\sqrt{\ell_p}\rceil)$ が与えられているとき，$[\vee_{j=1}^{\lceil\sqrt{\ell_p}\rceil} c_j]_{p'}$ を得る．$f(1) = 0$ かつ $f(2) = f(3) = \cdots = f(\lceil\sqrt{\ell_p}\rceil + 1) = 1$ となるような多項式 f を用意する．$d = (\sum_{j=1}^{\lceil\sqrt{\ell_p}\rceil} c_j) + 1$ として，$[f(d)]_{p'}$ を計算する．なお，$[f(d)]_{p'}$ の計算には積プロトコルを用いる．このとき，d は $1 \le d \le \lceil\sqrt{\ell_p}\rceil + 1$ の大小関係を持つ．

d の上限である $\lceil\sqrt{\ell_p}\rceil + 1$ が，Unbounded Fan-In Or ひいては BLT プロトコルにおける中間秘密情報の上限である．

表 1 JRNBS プロトコルの削減効果
Table 1 Effect on JRNBS protocols.

ℓ_p, $\ell_{p'}$	k	JRNBS	AJRNBS
$\ell_p = 32$, $\ell_{p'} = 4$	2	77,824	22,528 (28%)
$\ell_p = 32$, $\ell_{p'} = 4$	3	77,824	28,288 (36%)
$\ell_p = 64$, $\ell_{p'} = 5$	2	311,296	73,216 (23%)
$\ell_p = 64$, $\ell_{p'} = 5$	4	311,296	95,296 (31%)

Reveal

4.4 節で述べたように，p' の決定において，Reveal は考慮する必要がない．

以上より，法 p' を用いるステップ 1 から 3 のなかで，乱数の和積による値の秘匿および Reveal を除いた部分に着目すると，BLT プロトコルにおける中間秘密情報の上限 $\lceil\sqrt{\ell_p}\rceil + 1$ が最も大きい．そこで，AJRNBS において，p' が $\max(T, n) = \max(\lceil\sqrt{\ell_p}\rceil + 1, n)$ を超えるときに，正確性と安全性が維持される．この条件を ℓ_p の観点から書き換えると，$\ell_{p'} > \lfloor\log(\max(\lceil\sqrt{\ell_p}\rceil + 1, n))\rfloor + 1$ となる．なお，通信量の削減のため，p' は条件を満たす最も小さい素数に設定される．

5.3 通信量の削減効果

小さい法を用いることで，JRNBS のステップ 1 における通信量が削減される一方，CBBS の通信量が加わる．いくつかの ℓ_p および k に対する p' と $\ell_{p'}$ の値，そのときの JRNBS に対する AJRNBS の通信量を**表 1** に示す．

5.4 ラウンド数

AJRNBS は JRNBS と比べると，CBBS が必要になる分，ラウンドが増大する．ただし，CBBS のステップ 1 から 3 は 1 ビットの乱数 $[r_B]_p$ および $[r_B]_{p'}$ を生成するための処理であり，他の処理とは独立である．よって CBBS のステップ 1 から 3 は AJRNBS のステップ 1 から 2 までと並列に実行できる．一方，CBBS のステップ 4 は，AJRNBS のステップ 5 の結果 v を必要とする．したがって，CBBS 全体（4 ラウンド）のうち，3 ラウンドは並列化が可能な処理で，残る 1 ラウンドが並列化ができない処理となるため，AJRNBS は JRNBS と比べて，ラウンドが 1 増加し，8 ラウンドとなる．

5.5 計算量の増加

AJRNBS は，JRNBS に比べて法変換プロトコル（CBBS）ℓ_p 回分の計算量が増加し，それにともなう計算時間が加わる．本節では，CBBS による計算量の増加を見積もる．計算時間については，実装方法に依存するので，8.2 節にて実装を考慮した見積りを行う．

以下では，4.6 節で示したステップに沿って，参加者 1 の計算機の立場から，CBBS の計算量を見積もる．説明の都合上，ステップ 1，5，4，2，3，6 の順に分析する．

CBBS のステップ 1 において，参加者 1 の計算機は，1 ビットの乱数 $r_B.1$ を算出する．この乱数および本節の以下の部分で述べる乱数は，暗号学的に安全な乱数である必要がある．

次に，参加者 1 の計算機は，生成した 1 ビット乱数 $r_B.1$ を，素数 p と Shamir(k, n) 閾値秘密分散法により分散する．秘密分散は，3.1 節で述べたように，多項式 $f(x)$ を生成し，参加者$_i$ に対して $f(i)(1 \le i \le n)$ を算出する．$f(x)$ の生成では，係数となる乱数を $(k-1)$ 個生成する．この乱数は ℓ_p ビットである．

$f(i)$ の計算では，i^j の計算 $(1 \le i \le n, 1 \le j \le k-1)$ を行った後，i^j と係数との積を $(k-1)$ 回，積の結果および s の間の和を $(k-1)$ 回計算する．なお，i^j の計算は，複数回の CBBS さらには AJRNBS の他の秘密分散にも再利用できるので，1 回行えばよく，この計算量は無視してもいい．$f(i)$ の計算を 参加者$_i(1 \le i \le n)$ について行うので，合計で $(k-1)n$ 回の積，$(k-1)n$ 回の和を実行する．なお，ここでの積および和は，積プロトコルおよび和プロトコルではなく，$GF(p)$ 上の積および和である．これらの計算を以下では，単に積および和と呼ぶことにする．

参加者 1 の計算機は，素数 p' を用いた秘密分散も行うので，1 ビットの乱数 1 個の生成，$GF(p')$ 上の積 $(k-1)n$ 回および和 $(k-1)n$ 回がさらに必要である．

ステップ 5 では，Reveal プロトコルにより，1 ビットの値 t_B を復元する．Reveal では，k 個のシェアを引数とするラグランジュ補間を行う．ラグランジュ補間の計算の一部は，複数回の CBBS，さらには，AJRNBS の他の部分でも再利用できるので，1 回だけ計算すればよく，この計算量は無視できる．ラグランジュ補間のうち再利用できない部分の計算は，積 k 回，和 $(k-1)$ 回から構成される．

ステップ 4 は，素数 p で分散された 2 個のシェアの間で排他的論理和を計算する．この排他的論理和の計算は，積プロトコルに帰着する．積プロトコルの計算を以下に示す．

(1) 自己の持つ 2 つのシェア $[a]_p$ と $[b]_p$ について，$GF(p)$ 上の積を行う．結果を c とする．
(2) 値 c に Shamir(k, n) 閾値秘密分散法を適用し，n 個のシェアを生成する（生成したシェアを自分以外の $(n-1)$ 人の参加者の計算機に送る．一方，$(n-1)$ 人の参加者の計算機から，同様に生成されたシェアを受け取る）．
(3) 生成したシェアと受け取ったシェアのうち $(2k-1)$ 個を用いて，ラグランジュ補間により $[a \times b]_p$ を算出する．

このうち (2) における秘密分散の計算内容は前述のとおりである．(3) におけるラグランジュ補間は，$(2k-1)$ 個のシェアを引数とするので積 $(2k-1)$ 回，和 $(2k-2)$ 回となる．(1) から (3) の合計では，$GF(p)$ 上の積 $1+(k-1)n+(2k-1)$

回，和 $(k-1)n+(2k-2)$ 回，乱数生成 $(k-1)$ 回となる．

ステップ 2 では，素数 p で分散された k 個のシェアの間で排他的論理和を計算する．詳細は省略するが，この処理は Unbounded Fan-In Xor [9] を用いてラウンド数を削減している．その際の計算量は $GF(p)$ 上の積 $4k^2n-5kn+n+6k^2+2k-1$ 回，和 $4k^2n-3kn+n+6k^2-k-2$ 回，乱数生成 $4k^2-3\ell_p+1$ 回となる．同様に，ステップ 3 は $GF(p')$ 上の積 $4k^2n-5kn+n+6k^2+2k-1$ 回，和 $4k^2n-3kn+n+6k^2-k-2$ 回，乱数生成 $4k^2-3\ell_p+1$ 回となる．

ステップ 6 では，素数 p で分散されたシェアと平文の間で排他的論理和を計算する．$GF(p')$ 上の積 2 回，和 2 回を必要とする．

CBBS における乱数の総数は，1 ビット乱数 1 個，および ℓ_p ビット乱数 $4k^2-k-1$ 個と $\ell_{p'}$ ビット乱数が $4k^2-2k$ 個である．そのうちの ℓ_p ビット乱数は，秘密分散に用いるので，素数 p 未満でなければならない．そのため，生成した乱数と素数 p との大小比較を行い，条件を満たさない場合は乱数を棄却する．条件を満たす乱数を N 個生成するには，αN 回の乱数生成処理を実行する必要がある（$\alpha>1$）．（$\alpha=\frac{\text{発生確率 } p/(2^{\ell_p}-1) \text{ の事象が } N \text{ 回発生するまでの試行回数}}{N}$）であり，その期待値は p および N に依存して決まる．同様に，条件を満たす ℓ'_p ビットの乱数を N' 個生成するには，$\alpha' N'$ 回の乱数生成処理を実行する必要がある．

以上の CBBS の計算量を**表 2** にまとめる．AJRNBS において CBBS は ℓ_p 回実行される．

乱数生成は，暗号学的に安全な乱数を生成する必要がある．そのため，ストリーム暗号の乱数生成やブロック暗号 AES のカウンタモードによる乱数生成を用いる必要があり，積および和に比べて，1 回の計算量はきわめて大きい．現在 MPC の実用化が進められている分野は，データベースの秘密分散やクラウドサーバにおける顧客情報ファイルの秘密分散であり，これらの分野では膨大な通信を避けるために，(k,n) を小さく取る場合，たとえば $(k,n)=(2,3)$ または $(3,5)$ の場合が多い [14]．そこで，実用的な MPC を想定すると，たとえば，$(k,n)=(2,3)$ の場合には，乱数生成は，ℓ_p ビット 13 回，$\ell_{p'}$ ビット 8 回，1 ビット 1 回であり，積は，$GF(p)$ 上で 58 回，$GF(p')$ 上で 56 回，和は $GF(p)$ 上で 63 回，$GF(p')$ 上で 61 回であり，回数に大きな差はない．以上から，CBBS の計算のうち暗号学的乱数生成が支配的である．

表 2 CBBS の計算量
Table 2 Computational cost of CBBS.

処理	回数
1 ビット乱数生成	1 回
ℓ_p ビット乱数生成	$\alpha(4k^2-k-1)$ 回
$\ell_{p'}$ ビット乱数生成	$\alpha'(4k^2-2k)$ 回
$GF(p)$ 上の積	$4k^2n-3kn-n+6k^2+4k-1$ 回
$GF(p')$ 上の積	$4k^2n-3kn-n+6k^2+3k-1$ 回
$GF(p)$ 上の和	$4k^2n-1kn-n+6k^2+2k-4$ 回
$GF(p')$ 上の和	$4k^2n-1kn-n+6k^2+k-4$ 回

5.6 AJRNBS の正確性

5.1 節で述べたように，MJRNBS と JRNBS は，MJRNBS のステップ 5（リネーム）を除いて等価である．そこで，MJRNBS と AJRNBS の等価性を示すことで，JRNBS と AJRNBS の等価性を示す．MJRNBS および AJRNBS は，乱数のシェアを出力するプロトコルである．したがって，MJRNBS と AJRNBS の等価性とは，同じ値の乱数を出力することではなく，同じ性質すなわち，同じ値の範囲で同じ一様性を有する乱数を同じ個数生成することである．この観点から，MJRNBS と AJRNBS の等価性を示す．

以下では，ステップごとに分析する．ステップ 1 では，MJRNBS も AJRNBS も，JRBS を l_p 回実行している．3.8 節で述べたように，JRBS は，法に依存せず，$\{0,1\}$ 上で一様な 1 ビット乱数のシェアを生成する．したがって，MJRNBS も JRNBS も，ステップ 1 において，$\{0,1\}$ 上で一様な 1 ビット乱数のシェアを ℓ_p 個生成する．相違点は，シェアの法だけである．

ステップ 2 では，MJRNBS も AJRNBS も，BLT を実行する点は同じであるが，MJRNBS は法 p，AJRNBS は法 p' を用いる点が異なる．しかし，5.2 節の Unbouded Fan-In Or の説明で述べたように，BLT における中間秘密情報の上限は，$U=\lceil\sqrt{\ell_p}\rceil+1$ であり，p' は U よりも大きな値（$\max(U,n)$ を超える素数）に設定される．そのため，BLT の中間秘密情報の値が，$GF(p')$ の演算の影響を受けることはない．したがって，MJRNBS のステップ 2 と AJRNBS のステップ 2 は，同じ値の秘密を入力すると，同じ値の秘密を出力する．すなわち，MJRNBS も AJRNBS も，ステップ 2 において，$r<p$ の真偽を 1 ビットの値 v_B として出力し，秘密分散の法だけが異なる．

ステップ 3 と 4 を経て，ステップ 5 に進んだ時点で，MJRNBS も AJRNBS も，$r<p$ となる乱数 r のビットごとのシェアを持っている．より厳密には，$\{0,1\}$ 上で一様な 1 ビット乱数のシェア ℓ_p 個からなるベクトルであって，ℓ_p ビットの 2 進数と見なしたときに p より小さいベクトルを持っている．相違点は，秘密分散の法だけである．

AJRNBS のステップ 5 では，CBBS を用いて，法を p' から p に変える．CBBS は，秘密の値（シェアを復号したときの値）を変えずに，法だけを変える．そこで，AJRNBS のステップ 5 を経たときに，MJRNBS も AJRNBS も，$\{0,1\}$ 上で一様な 1 ビット乱数のシェア ℓ_p 個からなるベクトルであって，ℓ_p ビットの 2 進数と見なしたときに p より小さいベクトルを持っており，その法も等しい．

MJRNBS のステップ 5，6 と AJRNBS のステップ 6，7 は等価である．以上から，MJRNBS と AJRNBS は等価で

あり，どちらも，$\{0,1\}$ 上で一様な 1 ビット乱数のシェア ℓ_p 個を法 p の下で出力する．

5.7 AJRNBS の安全性

AJRNBS が JRNBS の安全性を維持しているかを考察する．ここで考察するべき安全性は 2 種類である．1 つ目は，JRNBS で漏洩しなかった秘密情報が AJRNBS でも漏洩しないことである．2 つ目は，JRNBS は乱数共有プロトコルであるから，JRNBS の出力する乱数の性質が，AJRNBS の出力する乱数でも維持されていることである．このうち 2 つ目については，上記の 5.5 節で述べたので，以下では情報漏洩がないことを示す．

JRNBS と MJRNBS の安全性は等しいので，MJRNBS で漏洩しなかった秘密情報が AJRNBS でも漏洩しないことを示す．AJRNBS が MJRNBS と異なる点は，ステップ 1 から 3 の秘密情報を小さい法 p' によって秘密分散している点と，ステップ 5 において CBBS を用いる点である．

以下では，ステップごとに分析する．ステップ 1 では，MJRNBS も AJRNBS も，JRBS を ℓ_p 回実行している．3.8 節で述べたように，JRBS の安全性（秘密が漏洩しないこと）は，法には依存しない．したがって，MRNBS と AJRNBS のステップ 1 の安全性は同等である．

ステップ 2 では，MJRNBS も AJRNBS も，BLT を実行する点は同じであるが，MJRNBS は法 p，AJRNBS は法 p' を用いる点が異なる．法が p であることは，秘密情報が 0 以上 $p-1$ 以下であることを示す．小さい法 p' の利用は，秘密情報が 0 以上 $p'-1$ 以下であることを示すので，秘密情報の範囲がより狭く推定される可能性がある．しかし，5.2 節の Unbounded Fan-In Or の説明で述べたように，BLT における中間秘密情報の上限は，$U=\lceil\sqrt{\ell_p}\rceil+1$ である．この上限の式は，BLT の仕様に基づき，5.2 節の分析によって求めたものである．この分析では，BLT のプロトコル仕様だけを用い，AJRNBS の個々の実行における秘密情報は用いていない．BLT のプロコル仕様は公開情報である．上限 U は，ℓ_p だけから算出可能であり，ℓ_p は MJRNBS における公開情報である．そこで，攻撃者は，MJRNBS において，公開された情報だけを用いて，BLT における中間秘密情報の上限 U を求めることができ，中間秘密情報の範囲が 0 以上 U 以下であることを知りうる．

以上をまとめると，攻撃者は，MJRNBS において，中間秘密情報の範囲が 0 以上 U 以下であることを知りうる．AJRNBS において，0 以上 $p'-1$ 以下であることを知りうる．p' は U よりも大きな値（$\max(U,n)$ を超える素数）に設定されるので，攻撃者は AJRNBS において，MJRNBS で知り得た以上の情報を知ることはできない．また，BLT の出力である v は，0 または 1 であるので，小さい法 p' で分散しても，値を推定することはできない．

ステップ 3 における Reveal の引数 $[v]_p$ の安全性については，上記のとおりである．最後に，AJRNBS のステップ 5 において CBBS を用いる点について述べる．4.6 節で述べたように，CBBS は秘密情報を漏洩しないので，ステップ 5 において秘密情報は漏洩しない．

以上から，MJRNBS で漏洩しなかった秘密情報は AJRNBS でも漏洩しない．

6. JRNBS 利用パターンの効率化

6.1 AJRNBS 利用パターン

小さい法 p' を用いて，JRNBS 利用パターンを効率化する．まず，4.5 節で示した JRNBS 利用パターンを下記の Modified-JRNBS 利用パターン（MJRNBS 利用パターン）に等価変形する．MJRNBS 利用パターンは，出力秘密情報の名前が u_B から t_B に変わっている以外，JRNBS 利用パターンと同じである．したがって，MJRNBS 利用パターンと JRNBS 利用パターンは，同じ入力秘密情報に対して同じ出力秘密情報を算出し，安全性も等しい．

MJRNBS 利用パターン

Input: $[s]_p$

1: $([r_1]_p,\cdots,[r_{\ell_p}]_p) \leftarrow$ JRNBS
2: $[r]_p \leftarrow ([r_1]_p,\cdots,[r_{\ell_p}]_p)$
3: $[r]_p$ を用いて s を秘匿した値 $[v]_p$ を計算．
4: $v \leftarrow \mathrm{Reveal}([v]_p)$
5: $[u_B]_p \leftarrow \mathrm{F}((v_1,\cdots,v_{\ell_p}),([r_1]_p,\cdots,[r_{\ell_p}]_p))$
6: $[u_B]_p$ を $[t_B]_p$ にリネーム
7: $[t_B]_p$ を出力

次に，MJRNBS 利用パターンに小さい法 p' を取り入れ，Advanced-JRNBS 利用パターン（AJRNBS 利用パターン）を構成する．MJRNBS 利用パターンの入力秘密情報は $[s]_p$ であり，出力秘密情報は $[t_B]_p$ である．中間秘密情報は，ステップ 1 から 2，およびステップ 4 から 5 の秘密情報である．ここでは，ステップ 5 における中間秘密情報を小さい法 p' によって秘密分散する．また，ステップ 1 においては，JRNBS の代わりに，AJRNBS を用いる．以下に，AJRNBS 利用パターンを示す．

AJRBS 利用パターン

Input: $[s]_p$

1: $([r_1]_p,\cdots,[r_{\ell_p}]_p),([r_1]_{p'},\cdots,[r_{\ell_p}]_{p'}) \leftarrow$ AJRNBS
2: $[r]_p \leftarrow ([r_1]_p,\cdots,[r_{\ell_p}]_p)$
3: $[r]_p$ を用いて s を秘匿した値 $[v]_p$ を計算．
4: $v \leftarrow \mathrm{Reveal}([v]_p)$
5: $[u_B]_{p'} \leftarrow \mathrm{F}((v_1,\cdots,v_{\ell_p}),([r_1]_{p'},\cdots,[r_{\ell_p}]_{p'}))$
6: $[u_B]_p \leftarrow \mathrm{CBBS}([u_B]_{p'})$
7: $[u_B]_p$ を $[t_B]_p$ にリネーム
8: $[t_B]_p$ を出力

AJRNBS 利用パターンのステップ 1 における AJRNBS は，5.1 節で述べた AJRNBS の出力秘密情報 $([r_1]_p,\cdots,[r_{\ell_p}]_p)$ 以外に，この出力秘密情報の法変換前の

秘密情報である $([r_1]_{p'}, \cdots, [r_{\ell_p}]_{p'})$（ステップ 5）を出力する（AJRNBS 利用ステップ 5）．この $([r_1]_{p'}, \cdots, [r_{\ell_p}]_{p'})$ を F の入力としてを用いる．

6.2 法 p' の決定

AJRNBS 利用パターンの法 p' を用いるビット演算は，ステップ 1（AJRNBS），4（ビット演算 F）である．5.2 節で述べたように，AJRNBS における秘密の中間値の上限は $\lceil\sqrt{\ell_p}\rceil + 1$ である．したがって，AJRNBS 利用パターンの法 p' を用いるビット演算において，秘密値の上限は，$\max(T_F, \lceil\sqrt{\ell_p}\rceil + 1, n)$ となる．ただし，T_F はビット演算 F における秘密値の上限である．以上から，AJRNBS 利用パターンにおいて，$p' > \max(T_F, \lceil\sqrt{\ell_p}\rceil + 1, n)$ を満たすとき，正確性と安全性が維持される．通信量の削減のため，p' は条件を満たす最も小さい素数に設定される．

6.3 通信量の削減効果

小さい法を用いることで，AJRNBS 利用パターンのステップ 1 と 4 における通信量が削減される一方，CBBS の通信量が加わる．具体的な通信量削減効果については，個々の具体的なビット演算 F に依存するため，7 節で明らかにする．

6.4 ラウンド数

AJRNBS 利用パターンと JRNBS 利用パターンの違いは，以下の 2 点である．この 2 点に着目し，2 つのパターンのラウンド数の違いを考える．

相違点 1：

AJRNBS 利用パターンは，ステップ 1 に JRNBS ではなく AJRNBS を用いている．JRNBS に比べ，AJRNBS は 1 ラウンド増大する．

相違点 2：

AJRNBS 利用パターンは，ステップ 5（CBBS）の処理が追加されている．ただし，CBBS（4 ラウンド）のうち，3 ラウンドは他とは独立した処理であり，ステップ 1（AJRNBS）の中の CBBS と並列して実行が可能である．したがって，ステップ 5 では 1 ラウンド増大する．

相違点により，AJRNBS 利用パターンは JRNBS 利用パターンと比べて，2 ラウンド増大する．

6.5 計算量の増加

AJRNBS 利用パターンは，JRNBS 利用パターンに比べて法変換（CBBS）の $(\ell_p + 1)$ 回分の計算量が増加する．その計算量の内訳は，5.5 節で述べたとおりである．

6.6 AJRNBS 利用パターンの正確性

6.1 節で述べたように，JRNBS 利用パターンと MJRNBS 利用パターンは．同じ入力秘密情報に対して，同じ出力秘密情報を算出する．そこで，MJRNBS 利用パターンと AJRNBS 利用パターンの等価性（同じ入力秘密情報に対して同じ出力秘密情報を算出すること）を示すことで，JRNBS 利用パターンと AJENBS 利用パターンの等価性を示す．

以下では，ステップごとに分析する．AJRNBS 利用パターンのステップ 1 と MJRNBS 利用パターンのステップ 1 を比べると，AJRNBS 利用パターンでは，JRNBS の代わりに AJRNBS を用いている．AJRNBS の出力秘密情報は $([r_1]_p, \cdots, [r_{\ell_p}]_p)$ であり，これは，5.5 節で述べたように，JRNBS の出力秘密情報 $([r_1]_p, \cdots, [r_{\ell_p}]_p)$ と等価である．AJRNBS 利用パターンのステップ 1 では，$([r_1]_p, \cdots, [r_{\ell_p}]_p)$ だけでなく，その法変換前の中間情報 $([r_1]_{p'}, \cdots, [r_{\ell_p}]_{p'})$ も出力している．AJRNBS の $([r_1]_{p'}, \cdots, [r_{\ell_p}]_{p'})$ と MJRNBS の $([r_1]_p, \cdots, [r_{\ell_p}]_p)$ が等価であることも，5.5 節で説明した．なお，ここでの等価性とは，$\{0,1\}$ 上で一様な 1 ビット乱数のシェア ℓ_p 個からなるベクトルであって，ℓ_p ビットの 2 進数と見なしたときに p より小さいベクトルという点において同じであることを意味する．

ステップ 2 では，AJRNBS 利用パターンも MJRNBS 利用パターンも，1 ビットの乱数のシェア $[r_i]_p (1 \le i \le \ell_p)$ 個から，ℓ_p ビットの乱数 $[r]_p$ を求める．AJRNBS 利用パターンの $[r_i]_p$ と MJRNBS 利用パターンの $[r_i]_p (1 \le i \le \ell_p)$ が等価であるため，AJRNBS 利用パターンの $[r]_p$ と MJRNBS 利用パターンの $[r]_p$ は等価であり，どちらも，0 以上 $p-1$ 以下の一様な乱数を法 p で分散したものである．そのため，AJRNBS 利用パターンと MJRNBS 利用パターンは，ステップ 3，4 において等価である．

AJRNBS 利用パターンのステップ 5 と MJRNBS 利用パターンのステップ 5 を比べると，AJRNBS が小さい法 p' を用いる点だけが異なり，他は同じである．6.2 節で述べたように，法 p' は，ビット演算 F における中間秘密情報の上限値 T_F より大きく設定される．そこで，$GF(p')$ 上の演算によって，ビット演算 F の中間秘密情報の値が影響を受けることはない．また，4.5 節で述べたように，F の出力 $[u_B]_p$ の値（復元したときの u_B の値）は，乱数の値にかかわらず，入力秘密情報の値だけに依存して決まる．以上から，入力秘密情報が等しければ，AJRNBS 利用パターンのステップ 5 の出力と MJRNBS 利用パターンのステップ 5 の出力は，同じ値のシェアであり，法が異なるだけである．

AJRNBS のステップ 6 は，u_B の値を変えずに，秘密分散の法だけを p' から p に変える．その結果，AJRNBS のステップ 6 の出力は，MJRNBS のステップ 5 の出力に比べて，値も法も等しい．AJRNBS のステップ 7，8 と MJRNBS のステップ 6，7 は等価である．

以上から，AJRNBS と MJRNBS は等価であり，同じ入力秘密情報に対して同じ出力秘密情報を出力する．

6.7 AJRNBS 利用パターンの安全性

MJRNBS 利用パターンで漏洩しなかった秘密情報が AJRNBS 利用パターンでも漏洩しないことを示す．以下では，ステップごとに分析する．AJRNBS 利用パターンのステップ 1 と MJRNBS 利用パターンのステップ 1 を比べると，AJRNBS 利用パターンでは，JRNBS の代わりに AJRNBS を用いている．5.6 節で示したように，AJRNBS の安全性（情報漏洩がないこと）は JRNBS に等しい．また，5.5 節で示したように，AJRNBS の出力する乱数 $([r_1]_p, \cdots, [r_{\ell_p}]_p)$ は，MJRNBS の出力する乱数 $([r_1]_p, \cdots, [r_{\ell_p}]_p)$ と等価である．

ステップ 2 では，AJRNBS 利用パターンも MJRNBS 利用パターンも，1 ビットの乱数のシェア $[r_i]_p (1 \leq i \leq \ell_p)$ 個から，ℓ_p ビットの乱数 $[r]_p$ を求める．AJRNBS 利用パターンの $[r_i]_p$ と MJRNBS 利用パターンの $[r_i]_p (1 \leq i \leq \ell_p)$ が等価であるため，AJRNBS 利用パターンの [r]p と MJRNBS 利用パターンの $[r]_p$ は等価であり，どちらも，0 以上 $p-1$ 以下の一様な乱数を法 p で分散したものである．したがって，ステップ 3，4 において，AJRNBS 利用パターンと MJRNBS 利用パターンの安全性は等しい．

AJRNBS 利用パターンのステップ 5 と MJRNBS 利用パターンのステップ 5 を比べると，AJRNBS が小さい法 p' を用いる点だけが異なり，他は同じである．小さい法 p' の利用は，秘密情報が 0 以上 $p'-1$ 以下であることを示すので，秘密情報の範囲がより狭く推定される可能性がある．しかし，法 p' は，ビット演算 F における中間秘密情報の上限値 T_F より大きく設定される．攻撃者は，MJRNBS 利用パターンにおいて，中間秘密情報の範囲が 0 以上 T_F 以下であることを知りうる．AJRNBS において，0 以上 $p'-1$ 以下であることを知りうる．p' は T_F よりも大きな値，すなわち $\max(T_F, \lceil\sqrt{\ell_p}\rceil + 1, n)$ を超える素数に設定されるので，攻撃者は AJRNBS 利用パターンにおいて，JRNBS 利用パターンで知り得た以上の情報を知ることはできない．

AJRNBS 利用パターンのステップ 6 では，CBBS を用いるが，4.6 節で述べたように，CBBS は秘密情報を漏洩しない．

以上から，MJRNBS 利用パターンで漏洩しなかった秘密情報は AJRNBS 利用パターンでも漏洩しない．

7. AJRNBS 利用パターンの適用

7.1 大小比較プロトコルへの適用

3.11 節で述べたように，大小比較プロトコルは，LSB プロトコルと積プロトコルからなり，全体の通信量は $279\ell_p^2 + 5\ell_p$，ラウンド数は 15 である [9]．ここでは，大小比較プロトコルのうち LSB プロトコルの部分に提案方式を適用する．LSB プロトコルの通信量は $93\ell_p^2 + \ell_p$ であり，大小比較プロトコル内で 3 回実行されるため，その通信量は $279\ell_p^2 + 5\ell_p$ である．LSB プロトコルのラウンド数は 13 である．LSB プロトコルの詳細を下記に示す．3.11 節では，LSB を用いて c の最下位ビットを算出する場合を説明したので，入力を $[c]_p$ としていたが，ここでは一般形を示すため，入力を $[s]_p$ とする．なお，▷ 以降は当該ステップの通信量を示す．

LSB プロトコル

Input: $[s]_p$

1: $([r_1]_p, \cdots, [r_{\ell_p}]_p) \leftarrow$ JRNBS ▷ $76\ell_p^2$
2: $[r]_p \leftarrow ([r_1]_p, \cdots, [r_{\ell_p}]_p)$
3: $[v]_p \leftarrow [s]_p + [r]_p$
4: $v \leftarrow \text{Reveal}([v]_p)$
5: $[u_B]_p \leftarrow \text{BLT}((v_1, \cdots, v_{\ell_p}), ([r_1]_p, \cdots, [r_{\ell_p}]_p))$ ▷ $17\ell_p^2$
6: $[c_B]_p \leftarrow v_1 \oplus [r_1]_p$
7: $[s_1]_p \leftarrow [u_B]_p \times (1 - [c_B]_p) + (1 - [u_B]_p) \times [c_B]_p$ ▷ ℓ_p
8: $[s_1]_p$ を出力

LSB を等価変形した Modified-LSB（MLSB）を以下に示す．MLSB は，ステップ 8 において，出力秘密情報をリネームする以外は，LSB と同じである．そこで，LSB と MLSB は同じ入力秘密情報に対して同じ出力秘密情報を算出し，その安全性は等しい．

Modified-LSB（MLSB）プロトコル

Input: $[s]_p$

1: $([r_1]_p, \cdots, [r_{\ell_p}]_p) \leftarrow$ JRNBS ▷ $76\ell_p^2$
2: $[r]_p \leftarrow ([r_1]_p, \cdots, [r_{\ell_p}]_p)$
3: $[v]_p \leftarrow [s]_p + [r]_p$
4: $v \leftarrow \text{Reveal}([v]_p)$
5: $[u_B]_p \leftarrow \text{BLT}((v_1, \cdots, v_{\ell_p}), ([r_1]_p, \cdots, [r_{\ell_p}]_p))$ ▷ $17\ell_p^2$
6: $[c_B]_p \leftarrow v_1 \oplus [r_1]_p$
7: $[s_1]_p \leftarrow [u_B]_p \times (1 - [c_B]_p) + (1 - [u_B]_p) \times [c_B]_p$ ▷ ℓ_p
8: $[s_1]p$ を $[t_B]_p$ にリネーム
9: $[t_B]_p$ を出力

MLSB のステップ 1，2，3，4 が，JRNBS 利用パターン（6.1 節）におけるステップ 1，2，3，4 に各々相当する．MLSB のステップ 5 から 7 が，JRNBS 利用パターンのステップ 5（ビット演算 F）に相当する．すなわち，ここでの F は，$(v_1, \cdots, v_{\ell_p})$ と $([r_1]_p, \cdots, [r_{\ell_p}]_p)$ を入力し，$[s_1]_p$ を出力する．AJRNBS 利用パターンを用いれば，LSB のステップ 5 から 7 を $GF(p')$ 上で実行できる．AJRNBS 利用パターンを LSB プロトコルに対して適用した Advanced LSB（ALSB）プロトコルを下に記す．上記の LSB における各ステップの通信量が，下記の ALSB における各ステップの通信量へと改善される一方，法変換（ステップ 8）の通信量が加わる．ALSB の通信量は，下記の ALSB の各ステップの通信量の合計であるため，$(5k+1)\ell_p^2 + (5k+95)\ell_p\ell_{p'} + (5k+1)\ell_p + (5k+3)\ell_{p'}$ である．ラウンド数は 6.5 節で述べたように 2 増加し，15 である．ALSB を利用した大小判定プロトコルの通信量は $(15k+3)\ell_p^2 + (15k+285)\ell_p\ell_{p'} + (5k+1)\ell_p + (5k+7)\ell_{p'}$

Advanced LSB（ALSB）プロトコル

Input: $[s]_p$

1: $([r_1]_p, \cdots, [r_{\ell_p}]_p), ([r_1]_{p'}, \cdots, [r_{\ell_p}]_{p'}) \leftarrow$ AJRNBS ▷ $(5k+1)\ell_p^2 + (5k+78)\ell_p\ell_{p'}$
2: $[r]_p \leftarrow ([r_1]_p, \cdots, [r_{\ell_p}]_p)$
3: $[v]_p \leftarrow [s]_p + [r]_p$
4: $v \leftarrow \mathrm{Reveal}([v]_p)$
5: $[u_B]_{p'} \leftarrow \mathrm{BLT}((v_1, \cdots, v_{\ell_p}), ([r_1]_{p'}, \cdots, [r_{\ell_p}]_{p'}))$ ▷ $17\ell_p\ell_{p'}$
6: $[c_B]_{p'} \leftarrow v_1 \oplus [r_1]_{p'}$
7: $[t_B]_{p'} \leftarrow (1 - [c_B]_{p'})[u_B]_{p'} + (1 - [u_B]_{p'})[c_B]_{p'}$ ▷ $\ell_{p'}$
8: $[t_B]_p \leftarrow \mathrm{CBBS}([t_B]_{p'})$ ▷ $(5k+1)\ell_p + (5k+2)\ell_{p'}$
9: $[t_B]_p$ を出力

であり，ラウンド数は 17 である．

p' の大きさを，6.2 節で示した中間秘密情報の上限を表す式 $U = \max(T_F, \lceil\sqrt{\ell_p}\rceil + 1, n)$ を用いて考える．ビット演算 F に相当する箇所は，ステップ 5 から 7 である．5.2 節で述べたように，ステップ 5（BLT）の中間秘密情報の上限は $\max(\lceil\sqrt{\ell_p}\rceil + 1, n)$ である．ステップ 6 と 7 は，1 ビットの秘密どうしの演算であるため，中間秘密情報の上限は $n = \max(1, n)$ である．ここから，F における中間秘密情報の上限は $T_F = \max(\lceil\sqrt{\ell_p}\rceil + 1, n)$ となる．p' は，$\max(\lceil\sqrt{\ell_p}\rceil + 1, n)$ を超える最小の素数に設定することができる．なお，計算量については，大小比較プロトコルは LSB を 3 回実行し，LSB の AJRNBS 利用パターンは CBBS を $(\ell_p + 1)$ 回実行するので，合計で CBBS の $3(\ell_p + 1)$ 回の計算量が増加となる．CBBS1 回分の計算量については，5.5 節で述べたとおりである．通信量の削減効果を**表 3** に記す．

7.2 その他のプロトコルへの適用

等号判定（EQ）プロトコルおよび区間判定（IT）プロトコルへの適用について述べる．ここでは，概略を述べ，詳細は付録 A.1 および A.2 にて述べる．

まず，等号判定プロトコル（3.12 節）への適用について説明する．等号判定プロトコルのステップ 1 から 4 が，JRNBS 利用パターンのステップ 1 から 4 に各々相当する．等号判定プロトコルのステップ 5 から 10 が，JRNBS 利用パターンのステップ 5（ビット演算 F）に相当し，$GF(p')$ 上の実行により，通信量を削減することができる．一方，法変換の通信量が増加分となる．通信量の削減効果を**表 4** に記す．なお，計算量については，CBBS の $(\ell_p + 1)$ 回分の計算量が増加となる．

次に，区間判定プロトコル（3.13 節）への適用について述べる．区間判定プロトコルのステップ 1 から 4 が，JRNBS 利用パターンにおけるステップ 1 から 4 に各々相当する．区間判定プロトコルのステップ 17 から 23 が，JRNBS 利用パターンのステップ 5（ビット演算 F）に相当し，$GF(p')$ 上の実行により，通信量を削減できる．一方，法変換の通信量が増加分となる．通信量の削減効果を**表 5** に記す．なお，計算量については，CBBS の $(\ell_p + 1)$ 回分の計算量が増加となる．

表 3 大小比較プロトコルの削減効果
Table 3 Effect on comparison protocols.

ℓ_p, $\ell_{p'}$	k	従来方式	提案方式
$\ell_p = 32$, $\ell_{p'} = 4$	2	285,856	74,532 (26%)
$\ell_p = 32$, $\ell_{p'} = 4$	3	285,856	91,992 (32%)
$\ell_p = 64$, $\ell_{p'} = 5$	2	1,143,104	236,757 (21%)
$\ell_p = 64$, $\ell_{p'} = 5$	3	1,143,104	303,342 (27%)

表 4 等号判定プロトコルの削減効果
Table 4 Effect on equality testing protocols.

ℓ_p, $\ell_{p'}$	k	従来方式	提案方式
$\ell_p = 32$, $\ell_{p'} = 7$	2	82,944	32,532 (40%)
$\ell_p = 32$, $\ell_{p'} = 7$	3	82,944	38,967 (47%)
$\ell_p = 64$, $\ell_{p'} = 8$	2	331,776	93,472 (28%)
$\ell_p = 64$, $\ell_{p'} = 8$	3	331,776	116,872 (35%)

表 5 区間判定プロトコルの削減効果
Table 5 Effect on interval testing protocols.

ℓ_p, $\ell_{p'}$	k	従来方式	提案方式
$\ell_p = 32$, $\ell_{p'} = 4$	2	112,672	27,284 (24%)
$\ell_p = 32$, $\ell_{p'} = 4$	3	112,672	33,224 (29%)
$\ell_p = 64$, $\ell_{p'} = 5$	2	450,592	84,865 (19%)
$\ell_p = 64$, $\ell_{p'} = 5$	3	450,592	107,290 (24%)

8. 実装時の効率に関する考察

8.1 概要

本節では，従来方式と提案方式の実装時の効率を，計算，記憶，通信の面から考察する．提案方式の基本は，JRNBS の効率化であることから，従来プロトコルとして JRNBS，提案プロトコルとして AJRNBS を取り上げる．

なお，3.2 節で述べたように，MPC の基本プロトコルは，積プロトコル，和プロトコル，乱数生成プロトコル，Reveal プロトコルの 4 つであり，JRNBS などの他のプロトコルは，4 つの基本プロトコルから構成される．

8.2 計算量および計算時間の考察

本節では，JRNBS および AJRNBS の計算量を見積もり，計算量の支配的な要因（全体の大部分を占める計算）が乱数生成であることを明らかにする．次に，計算時間について考察する．計算時間は，下記の理由により，計算量に比例しない可能性がある．

(1) ソフトウェアとして実装する場合には，プログラミングによっては，無駄が発生する．たとえば，C 言語で実装する場合，5 ビットデータの処理に 8 ビットの文字型変数を用いれば，8 ビットデータの処理と同じ計算時間になる．アセンブラプログラミングなどにより，この無駄をなくせる可能性はあるが，プログラミ

ングの生産性および保守性が大きく損なわれる．そのため，一般的なプログラム言語を用い，処理効率と生産性・保守性のバランスを考慮しながらプログラミングを行うことが多い．

(2) レジスタなどが 32 ビット，64 ビット単位でしか処理できないことから，データが小さくなっても計算時間は変わらない可能性がある．たとえば，レジスタ長が 32 ビットの場合，データが 32 ビットでも 5 ビットでも，1 回の演算におけるレジスタの使用回数は等しい．

そこで，支配的な計算（乱数生成）を取り上げ，上記の (1) について，実際のプログラミングを想定して考察する．最後に，乱数生成を実装し，処理時間を測定することで，上記 (2) の影響を含めた JRNBS と AJRNBS の処理時間の比較を行う．

まず，JRNBS の計算量を見積もる．JRNBS（3.10 節参照）は，JRBS，BLT，Reveal からなる．JRBS（3.8 節）は乱数生成プロトコル JRNS（3.5 節）を含む．これらのプロトコル内の基本プロトコルの回数を合計すると，JRNBS は $48\ell_p$ 回の積プロトコル，$20\ell_p$ 回の和プロトコル，$28\ell_p+4$ 回の Reveal プロトコル，$28\ell_p$ 回の乱数生成プロトコルを含む．

積プロトコル 1 回分の計算の内訳は，5.5 節で述べたように，積 $1+(k-1)n+(2k-1)$ 回，和 $(k-1)n+(2k-2)$ 回，乱数生成 $(k-1)$ 回である．なお，ここでの積および和は，積プロトコルおよび和プロトコルではなく，$GF(p)$ 上の積および和である．また，乱数は，暗号学的に安全な乱数である必要がある．

和プロトコル 1 回の計算の内訳は，$GF(p)$ 上の和 1 回である．Reveal の計算の内訳はラグランジュ補間であり，ラグランジュ補間の内訳は，5.5 節で述べたように，$GF(p)$ 上の積 k 回，和 $(k-1)$ 回である．乱数生成プロトコルの計算の内訳は，積 $n(k-1)$ 回，和 $n(k-1)$ 回，乱数生成 k 回である．

以上述べた 4 つの基本プロトコルの回数および，基本プロトコルの内訳に基づき，JRNBS における乱数生成，積，和の回数を**表 6** にまとめる．なお，5.5 節で述べたように，積プロトコルの中で生成する乱数は，p 未満でなければならないので，$\beta(k-1)$ 個の乱数生成が必要である（$\beta>1$）．

5.5 節で述べたように，乱数生成は暗号処理を必要とし，他の計算に比べてきわめて計算量が大きい．また，現在 MPC の実用化が検討されている分野では，(k,n) を小さく取る場合，たとえば $(k,n)=(2,3)$ または $(3,5)$ の場合が多い [14]．そこで，実用的な MPC を想定すると，たとえば，$(k,n)=(2,3)$ かつ $\ell_p=32$ の場合には，乱数生成は 3,328 回，積は 15,240 回，和 11,908 回であり，回数に大きな差はない．以上から，全体の計算量のうち乱数生成が支配的である．

表 6 JRNBS の計算量
Table 6 Computational cost of JRNBS.

処理	回数
ℓ_p ビット乱数生成	$\beta(76\ell_p k-48\ell_p)$ 回
$GF(p)$ 上の積	$\ell_p(76kn-76n+124k)+4k$ 回
$GF(p)$ 上の和	$\ell_p(76kn-76n+124k-104)+4k-4$ 回

AJRNBS についても，同様の分析を行った．AJRNBS は，法変換プロトコル CBBS を ℓ_p 回実行する点が，JRNBS からの増加分となる．しかし，5.5 節で述べたように，CBBS でも乱数生成が支配的である．そこで，AJRNBS でも，JRNBS と同様に，乱数生成が支配的である．

AJRNBS の CBBS 以外の部分は，JRNBS と同じ計算であり，相違点は，和と積が $GF(p')$ 上である点，および乱数が $\ell_{p'}$ ビット乱数である点にある．CBBS 1 回の乱数生成は，1 ビット乱数 1 個，ℓ_p ビット乱数 $\alpha(4k^2-k-1)$ 個，$\ell_{p'}$ ビット乱数 $\alpha'(4k^2-2k)$ 個であり，CBBS は ℓ_p 回実行される．したがって，CBBS を含めた AJRNBS の乱数生成は，1 ビット乱数生成が 1 回，ℓ_p ビット乱数生成が $\gamma(4k^2\ell_p+75\ell_p k-49\ell_p)$ 回，$\ell_{p'}$ ビット乱数 $\gamma'(4k^2\ell_p+74l\ell_p-48\ell_p)$ 個（$\gamma>1,\gamma'>1$）となる．

次に乱数生成のプログラミングについて考察する．乱数生成には，Open SSL [15] で提供されている AES をカウンタモードで用いることにした．カウンタモードの AES は暗号学的に安全な乱数を生成することが知られている [8]．JRNBS（あるいは AJRNBS）で必要なすべての乱数を一括して生成することにした．JRNBS および AJRNBS で必要な乱数の個数およびビット数は，上記のとおりである．AES は 128 ビット単位で乱数を生成するので，基本的に，AES を ⌈必要な乱数のビット数の合計/128⌉ 回実行する．

AES 以外は C 言語を用いて実装することにした．$\ell_p=32,64$ の場合は，AES の生成した乱数から ℓ_p ビットごとに切り出し，p 未満であるかを判定する．$\ell_{p'}=4$ の場合には，文字列型（8 ビット）単位で切り出し，4 ビットを上下 2 個用いる．$\ell_{p'}=5$ の場合には，short int 型（16 ビット）単位で切り出し，5 ビットを 3 個用いることにしたので，AES の生成する乱数は 16/15 倍に増加することになった．ℓ_p および $\ell_{p'}$ ビットの乱数の切り出し，および素数 p との比較を，p 未満の乱数が必要な個数得られるまで繰り返すことにした．

以上の設計のとおり，JRNBS および AJRNBS の乱数生成を実装し，3.4 GHz，4 コアの CPU，64 ビットレジスタおよび 8 GB の主記憶を有する計算機上で処理時間を測定した．JRNBS の場合は，$\ell_p=32,64$ とし，AJRNBS の場合は，$(\ell_p,\ell_{p'})=(32,4),(64,5)$ とした．なお，γ，γ' は，素数 p，p' が大きいほど小さくなり，素数が小さいほど大きくなる．そこで，各々の ℓ_p および $\ell_{p'}$ について，最大の p および p' を用いた．具体的には，32 ビット素数として 0xfffffffb，64 ビット素数として 0xffffffffffffffc5，4 ビット素数として

表 7 乱数生成の計算時間
Table 7 Compution time of random number generation.

JRNBS		AJRNBS	
ℓ_p	時間 (s)	$(\ell_p, \ell_{p'})$	時間 (s)
32	0.000120	$(32, 4)$	0.00004418
64	0.000119	$(64, 5)$	0.000140

表 8 AJRNBS の記憶容量
Table 8 Storage requirement for AJRNBS.

JRNBS		AJRNBS	
ℓ_p	記憶容量 (bit)	$(\ell_p, \ell_{p'})$	記憶容量 (bit)
32	147,456	$(32, 4)$	81,920
64	589,824	$(64, 5)$	304,640

13，5 ビット素数として 31 を用いた．$(k, n) = (2, 3)$ とした．以上の測定結果を**表 7** に示す．表内の数値は各々 1,000 回測定した平均値である．

8.3 記憶容量の考察

一般にプログラム実行時には，プログラムを格納するための記憶容量と，変数や定数などのデータを格納する記憶容量が必要となる．従来のプロトコルの扱うデータは ℓ_p ビットであり，提案プロトコルの扱うデータは $\ell_{p'}$ ビットであることから，提案方式の影響を受けるのはデータの記憶容量である．そこで，データの記憶容量について分析する．原理的には，提案プロトコルの必要とする記憶容量は従来プロトコルに比べて $\ell_{p'}/\ell_p$ になる．しかし，計算量の考察でも述べたように，ソフトウェアとして実装する場合には，無駄が発生する可能性があり，原理どおりに記憶容量が削減できるとは限らない．さらに，提案プロトコルは，従来プロトコルの用いなかった CBBS を用いるため，CBBS のための記憶容量も考慮する必要がある．これらの点をふまえて，従来プロトコル JRNBS と提案プロトコル AJRNBS の記憶容量について考察する．

2 つの ℓ_p ビットの値の間で $GF(p)$ 上の演算を行う場合，処理の途中で最大 $2\ell_p$ ビットの値が生じる．そのため，本節の分析では，MPC を法 ℓ_p（あるいは $\ell_{p'}$）で実行する部分については，プログラムの変数および定数について，各々 $2\ell_p$（あるいは $2\ell_{p'}$）ビットの記憶容量を確保すると考える．より正確には，一般的なプログラム言語の変数，定数として指定可能なビット数であって，$2\ell_p$ ビット以上の最小の記憶容量を確保する．この記憶容量を $m(\ell_p)$ で表すことにする．たとえば，C 言語を用いる場合，$m(32) = 64$，$m(5) = 16$ となる．

JRNBS（3.10 節参照）のうち最も記憶容量を要する部分は，ステップ 1 において，JRBS を $4\ell_p$ 並列で実行するときであり，なかでも，JRBS（3.8 節）のステップ 2 において，$[r]_p \times [r]_p$ の積プロトコルを実行するときである．積プロトコルでは，秘密分散法により n 個のシェアを生成する．そのとき，$k-1$ 個の乱数と n 個のシェアおよび分散前の値 1 個を記憶する必要がある．$4\ell_p$ 並列の積プロトコルでは，ℓ_p ビットの値 $4\ell_p(k+n)$ 個記憶する必要がある．さらに，8.1 節で述べたように，JRNBS で必要とする乱数を一括生成する場合には，ℓ_p ビットの乱数を $\beta(76\ell_p k - 48\ell_p)$ 個記憶する必要がある．以上から，記憶容量は $(76\ell_p k - 48\ell_p) \times \ell_p + (4\ell_p(k+n)) \times m(\ell_p)$ となる．

AJRNBS（5.1 節参照）のうち，CBBS を除く部分で，最も記憶容量を要するのは，JRNBS を同じく JRNBS のステップ 1 であり，記憶容量は $(76\ell_p k - 48\ell_p) \times \ell_{p'} + (4\ell_p(k+n)) \times m(\ell_{p'})$ となる．AJRNBS では，CBBS を用いる．CBBS のうち最も記憶容量を要する部分は，ステップ 2 と 3 の並列実行において，k 個の ℓ_p ビット乱数の排他的論理和と k 個の $\ell_{p'}$ ビット乱数の排他的論理和を計算する部分である．5.5 節で述べたように，この計算は，Unbounded Fan-In Xor [6], [9] を用いて効率化することが可能である．詳細は省略するが，この部分の記憶容量は $(4k^2 - 2k) \times m(\ell_p) + (4k^2 - 2k) \times m(\ell'_p)$ という式となる．AJRNBS は，ℓ_p 回の CBBS を並列実行するので，記憶容量は，$\ell_p((4k^2 - 2k) \times m(\ell_p) + (4k^2 - 2k) \times m(\ell'_p)) + (76\ell_p k - 48\ell_p) \times \ell_{p'} + (4\ell_p(k+n)) \times m(\ell_{p'})$ となる．この CBBS のステップ 2，3 の記憶容量は，上述した JRBS のステップ 1 の記憶容量より大きいので，これが AJRNBS の記憶容量となる．

$\ell_p = 32$ と 64 における JRNBS の記憶容量，$(\ell_p, \ell_{p'}) = (32, 4), (64, 5)$ における AJRNBS の記憶容量を**表 8** に示す．$k = 2$，$n = 3$ とした JRNBS と AJRNBS の記憶容量を比較すると，JRNBS の方が大きい．その理由は CBBS を用いることで乱数における記憶容量が削減できることにある．

8.4 通信量および通信時間の考察

5.3 節および 7 章では，提案手法による通信量の削減効果の理論値を示した．しかし，以下の理由から，実装時の通信時間は，理論値どおりの削減にはならない．

(1) ソフトウェアとして実装する場合，プログラミングによっては，通信量が増加する可能性がある．
(2) 一般的に用いられる IP プロトコルの場合，通信データはパケットとして送受信される．個々のパケットには，通信データ量とは無関係に一定長のヘッダが付与される．
(3) レジスタなどが 32 ビット，64 ビット単位でしか処理できないことから，データ長がレジスタ長の倍数でない場合に，無駄な処理が発生する可能性がある．
(4) パケットの送受信では，送信側でのパケットの組み立て（通信データへのヘッダの付与など）や受信側でのシーケンス番号の確認のように，パケットのサイズと

は無関係の処理時間が存在する．

(5) OS の割り込みなどの外乱要因，およびネットワークカードなどの製品の特性の影響がある．

そこで，これらの点をふまえて，従来プロトコル JRNBS と提案プロトコル AJRNBS の通信時間について評価する．なお，ℓ_p として 32 と 64，$\ell_{p'}$ として 4 と 5 を取り上げる．

まず，(1) について考察する．JRNBS は，4 つの基本プロトコル（積プロトコル，和プロトコル，乱数生成プロトコル，Reveal プロトコル）の組合せになっている．JRNBS は 7 ラウンドからなり，各ラウンドにおいて，複数の基本プロトコルを並列で実行する．この並列実行されるプロトコルの通信データは 1 つの大きなデータにまとめて送受信することができる．たとえば，あるラウンドで N 回の積プロトコルを並列実行し，各々の積プロトコルは $\ell_p = 32$ ビットのデータを通信する場合，32 ビットデータ N 個をまとめて，$32N$ ビットのデータを 1 回通信すればよい．そのとき，送信側では，$32N$ ビットの配列（たとえば C 言語における長さ N の整数型配列）を設け，N 個の積プロトコルが，配列の異なるアドレスに 32 ビットのデータを書き込み，受信側では逆に 32 ビットデータを読み出せばよい．したがって，$\ell_p = 32, 64$ の場合には，通信量の無駄はなく，上記 (1) の問題は生じない．

$\ell_{p'} = 4$ とし，4 ビットデータ N 個をまとめて送る場合を考える．この場合は，たとえば長さ $\lceil N/2 \rceil$ の文字型配列を用意し，2 つの積プロトコルの通信データを，1 つの文字型データの上位 4 ビットと下位 4 ビットに書き込む．N が奇数の場合には，最後の文字型変数のうち 4 ビットが無駄になるが，この無駄は小さいため無視してもよい．このような実装を行うと，データを 4 ビットシフトするなどの計算が発生する．しかし，本節で後述するように，MPC の処理時間においては，計算よりも通信の方が支配的であるため，計算を増やして，通信量の増加を抑えることは不自然でない．5 ビットデータ N 個すなわち $5N$ ビットをまとめて送る場合には，たとえば，C 言語における長さ $\lceil N/3 \rceil$ の short int 配列を設け，5 ビットシフトにより，1 つの short int 変数に 3 個のデータを書き込む．この場合，short int 変数の 16 ビットのうち 1 ビットは無駄になり，最後の変数については，1，6，11 ビットのいずれかの無駄になるので，全体として，通信量が $\frac{16\lceil N/3 \rceil}{5N}$ 倍になる．

表 9 は，JRNBS および AJRNBS のラウンドごとの通信量の理論値（ビット数）を，$(\ell_p, \ell_{p'}) = (32, 4)$ および $(64, 5)$ の場合に示す．たとえば 32 ビットのシェアを用いる JRNBS では，1 ラウンド目で 28,672 ビットのデータを送信し，2 ラウンド目で 16,384 ビットのデータを送信する．$(\ell_p, \ell_{p'}) = (64, 5)$ の場合の AJRNBS では，通信データの一部は 5 ビットであるため，上記 (1) の要因により，通信量が増加する．そこで，上記 (1) の要因を考慮した AJRNBS の通信量を，表の右側の列に併記した．

表 9 通信量の理論値
Table 9 Theoretical communication amount.

ラウンド	JRNBS		AJRNBS		
	ℓ_p bit		$(\ell_p, \ell_{p'})$ bit		
	32	64	$(32, 4)$	$(64, 5)$	$(64, 5)$
1	28,672	114,688	10,176	41,636	41,676
2	16,384	65,536	5,376	23,672	23,692
3	8,192	32,768	9,920	41,236	41,249
4	8,192	32,768	128	200	213
5	4,096	16,384	64	100	106
6	8,192	32,768	128	200	213
7	4,096	16,384	64	100	106
8			128	320	320

表 10 通信の計測時間
Table 10 Measured communication time.

JRNBS		AJRNBS	
ℓ_p	時間 (ms)	$(\ell_p, \ell_{p'})$	記憶容量 (ms)
32	17.44	$(32, 4)$	17.95
64	35.51	$(64, 5)$	26.80

1 G ビットの LAN において，表 9 の通信時間を測定した結果を**表 10** に示す．使用した計算機は計算時間の測定に用いたものと同じである．1,000 回の測定を 1 セットとして，異なる時間帯で 3 セット測定し，3,000 で割って平均時間を求めた．表 10 を見ると，JRNBS と AJRNBS の通信時間にはほとんど差がなく，提案方式の効果は見られない．これは，上記の要因 (2) から (5) の影響であると考えられる．なお，表 10 の通信時間を，8.2 節の表 7 の計算時間と比べると，MPC においては，通信時間が支配的であることが分かる．

MPC を実用化する場合には，同じあるいは異なる複数のプロトコルを並列実行する場面が多数考えられる．たとえば，文字列探索において，Doc=“This is a pen” という文字列が秘密分散されており，その中から，Term=“is” という秘密分散された文字列を探索する場面を考える（実際には通信文からキーワードを探索するなどの応用がある）．この場面では，Doc の各部分すなわち，“Th”，“hi”，“is”，$\cdots$ “en” と Term=“is” との等号判定プロトコルを並列で実行するので，12 並列となる．一般には，（Doc の文字数—Term の文字数）の並列数となる．また，秘密分散された時系列データ a_i と b_i （$1 \le i \le n$，i は時刻 t_i の番号）の大小比較の場合，大小比較プロトコルの n 並列となる．たとえば，a_i が b_i よりつねに大きいかの判定，a_i が b_i より大きい時刻を求めるなどの場合がある．たとえば，等号判定プロトコルは JRNBS を用いるので，上記の文字列探索の場合は JRNBS の 12 並列となる．また，大小比較プロトコルは JRNBS を 3 並列で用いるので，上記の時系列データの比較の場合は，JRNBS の $3n$ 並列となる．

JRNBS および AJRNBS の並列数を M とすると，**表 11**

表 11 並列化した通信の計測時間
Table 11 Measured communication time of parallel processes.

並列数（M）	JRNBS		AJRNBS	
	ℓ_p bit		$(\ell_p, \ell_{p'})$ bit	
	32	64	(32, 4)	(64, 5)
1	17.44	35.51	17.95 (103%)	26.80 (75%)
10	96.88	518.47	67.86 (70%)	245.76 (48%)
50	538.63	2,665.48	253.81 (47%)	813.47 (31%)
100	1,123.31	5,156.85	411.93 (27%)	1,188.6 (23%)

に示したラウンドごとの通信データ量は M 倍になる．表 11 は，並列数 M = 10, 50, 100 の場合の通信時間を示す．この表は，並列数が大きくなると，AJRNBS による通信時間の低減効果が理論値に近づくことを示している．

$\ell_p = 32$ ビットかつ $\ell_{p'} = 4$ ビットの AJRNBS を 1 回並列すると，処理の終了まで 17.95 ms の通信時間を要する．これは，同条件，同並列数の JRNBS の通信時間（17.44 ms）のおよそ 103%の通信時間である．しかし，並列数が増大するにつれて，JRNBS と AJRNBS の通信時間の割合は逆転し，100 並列では，JRNBS の通信時間が 1,123.31 ms であるのに対して，AJRNBS の通信時間は 411.93 ms となる．100 並列では AJRNBS の通信時間は JRNBS の通信時間の 27%である．これは表 1 に示した削減効果の理論値（28%）と近い値である．

以上から，提案方式は，プロトコルを単独で実行する場合には通信時間削減の効果は小さいが，複数のプロトコルを並列実行する場合には効果があり，並列数が大きくなるに従って，削減効果が理論値に近づくと考えられる．

9. まとめ

秘密情報を少ないビット数で表現し，通信量を低減する手法を提案した．MPC の多くの基本的なプロトコルは，内部にビット演算を含む．そこで，多くのプロトコルに共通する 2 種類のビット演算を見い出した．1 つは，ビット単位での乱数の共有（Joint Random Number Bitwise-Sharing：JRNBS）であり，もう 1 つは，JRNBS を利用して，秘密情報のシェアから 1 ビットのシェアを算出する演算である．プロトコルの入出力情報が法 p で分散されている場合に，提案方式は法変換を行うことで，$p' > \max(T', \lceil\sqrt{\ell_p}\rceil + 1, n)$ となるような小さな法 p' でビット演算部分を実行する．ただし，T' はビット演算における秘密値の中間値の上限であり，n は MPC における参加者数である．

提案方式では，通信量を削減しながら，元のプロトコルの正確性と安全性を維持することができる．この手法により，(2, 3) 閾値法で法 p が 64 ビットの場合，大小比較，等号判定，区間判定の通信量を各々 80 パーセント，70 パーセント，80 パーセント程度削減できる．

参考文献

[1] Aleksandr, Y.: Efficient Cryptographic Tools for Secure Distributed Computing, Ph.D. Dissertation at the Graduate School of Yale University (2006).

[2] Bar-Ilan, J. and Beaver, D.: Non-cryptographic fault-tolerant computing in a constant number of rounds of interaction, *Proc. 8th Annual ACM Symposium on Principles of Distributed Computing*, pp.201–209, ACM (1989).

[3] Ben-Or, M., Goldwasser, S. and Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation, *20th Annual ACM Symposium on Theory of Computing*, pp.1–10, ACM (1988).

[4] Burkhart, M., Strasser, M., Many, D. and Dimitropoulos, X.: SEPIA: Privacy-preserving aggregation of multi-domain network events and statistics, *19th USENIX Security Symposium*, p.15 (2010).

[5] Cramer, R. and Damgård, I.: Secure distributed linear algebra in a constant number of rounds, *Proc. CRYPTO 2001*, Kilian, J. (Ed.), *LNCS*, Vol.2139, pp.119–136, Springer (2001).

[6] Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J.B. and Toft, T.: Unconditionally Secure Constant-Rounds Multi-party Computation for Equality, Comparison, Bits and Exponentiation, *TCC 2006, LNCS*, Vol.3876, pp.285–304 (2006).

[7] Gennaro, R., Rabin, M.O. and Rabin, T.: Simplified VSS and fast-track multiparty computations with applications to threshold cryptography, *17th Annual ACM Symposium on Principles of Distributed Computing*, pp.101–111, ACM (1988).

[8] Lipmaa, H., Rogaway, P. and Wagner, D.: Comments to NIST concerning AES modes of operation: CTR-mode encryption (2000), available from ⟨http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ctr/ctr-spec.pdf⟩.

[9] Nishide, T. and Ohta, K.: Multiparty Computation for Interval, Equality, and Comparison Without Bit-Decomposition Protocol, *PKC 2007, LNCS*, Vol.4450, pp.343–360 (2007).

[10] SecureSCM: Security Analysis Technical Report D9.2, available from ⟨http://www.securescm.org⟩ (2009).

[11] Shamir, A.: How to Share a Secret, *CACM*, Vol.22, No.11, pp.612–613 (1979).

[12] Schoenmaker, B. and Tuyls. P.: Efficent binary conversion for Paillier encrypted values, *Proc. Eurocrypt 2006, LNCS*, Vol.4004, pp.522–537, Springer-Verlag (2006).

[13] 加藤 遼，吉浦 裕：小さい法を用いたセキュアマルチパーティ計算のビット演算の効率化，コンピューターセキュリティシンポジウム 2013 (2013).

[14] 千田浩司，五十嵐大，濱田浩気，高橋克己：エラー検出可能な軽量 3 パーティ秘匿関数計算の提案と実装評価，情報処理学会論文誌，Vol.52, No.9, pp.2674–2685 (2011).

[15] Open SSL (online), available from ⟨https://www.openssl.org⟩.

付 録

A.1 AJRNBS 利用パターンの等号判定プロトコル (EQ) への適用

AJRNBS，AJRNBS 利用パターンおよび ALSB の場合は，MJRNBS，MJRNBS 利用パターンおよび MLSB を経

由して間接的に導出したが，同様の間接的な導出を行うと，説明が長大になるため，直接的な導出を行う．3.12 節で述べたように，等号判定プロトコル（文献 [9]）の通信量は $81\ell_p^2$，ラウンド数は 8 である．等号判定プロトコルのステップごとの通信量を ▷ 以降に示す．

等号判定プロトコル

Input: $[a]_p$, $[b]_p$
1: $([r_1]_p, \cdots, [r_{\ell_p}]_p) \leftarrow$ JRNBS ▷ $76\ell_p^2$
2: $[r]_p \leftarrow ([r_1]_p, \cdots, [r_{\ell_p}]_p)$
3: $[v]_p \leftarrow [a]_p - [b]_p + [r]_p$
4: $v \leftarrow \mathrm{Reveal}([v]_p)$
5: **for** $i := 1 \cdots \ell_p$ **do**
6: **if** $v_i = 0$ **then**
7: $[c_i]_p \leftarrow 1 - [r_i]_p$
8: **else**
9: $[c_i]_p \leftarrow [r_i]_p$
10: **end if**
11: **end for**
12: $[t_B]_p \leftarrow \mathrm{UnboundedFanInAnd}(([c_1]_p, \cdots, [c_{\ell_p}]_p))$ ▷ $5\ell_p^2$
13: $[t_B]_p$ を出力

等号判定プロトコルのステップ 1 から 4 が，JRNBS 利用パターンにおけるステップ 1 から 4 に各々相当する．等号判定プロトコルのステップ 5 から 12 が，JRNBS 利用パターンのステップ 5（ビット演算 F）に相当する．すなわち，F は等号判定プロトコルのステップ 6 で $(v_1, \cdots, v_{\ell_p})$ を入力し，ステップ 7 で $([r_1]_p, \cdots, [r_{\ell_p}]_p)$ を入力して，ステップ 12 で $[t_B]_p$ を出力する．AJRNBS 利用パターンを用いれば，等号判定プロトコルのステップ 5 から 12 を $GF(p')$ 上で実行できる．AJRNBS 利用パターンを等号判定プロトコルに対して適用した Advanced EQ（AEQ）プロトコルを下記に示す．上記の等号判定プロトコルにおける各ステップの通信量が，下記の AEQ プロトコルにおける各ステップの通信量へと改善される一方，法変換（ステップ 13）の通信量が加わる．AEQ の通信量は，下記の AEQ の各ステップの通信量の合計であるため，$(5k+1)\ell_p^2 + (5k+83)\ell_p\ell_{p'} + (5k+1)\ell_p + (5k+2)\ell_{p'}$ である．ラウンド数は 6.5 節で述べたように 2 増加し，10 である．

p' の大きさを，6.2 節で中間秘密情報の上限を表す式 $\max(T_F, \lceil\sqrt{\ell_p}\rceil + 1, n)$ を用いて考える．ビット演算 F に相当する箇所は，ステップ 5 から 12 である．そのうち，ステップ 5 から 12 は 1 ビットの秘密どうしの演算であり，上限は $n = \max(1, n)$ である．ステップ 12 の Unbounded Fan-In And は 6.2 節で示した Unbounded Fan-In Or の一部を変更したものである．Unbounded Fan-In Or は ℓ_p 個のビット値を $\sqrt{\ell_p}$ 個ずつ区切って処理していたため，その値の上限は $\max(\lceil\sqrt{\ell_p}\rceil + 1, n)$ であった．Unbounded Fan-In And も同様であるが，ℓ_p 個のビット値を区切らずに処理するため，上限は $\max(\ell_p + 1, n)$ である．ここから，F における中間秘密情報の上限は $T' = \max(\ell_p + 1, n)$ となる．ここから，F における中間秘密情報の上限は $T_F = \max(\ell_p + 1, n)$ となる．p' は，$\max(\ell_p + 1, n)$ を超える最小の素数に設定することができる．通信量の削減効果および計算量の増加については，7.2 節で述べたとおりである．

Advanced 等号判定（AEQ）プロトコル

Input: $[a]_p$, $[b]_p$
1: $([r_1]_p, \cdots, [r_{\ell_p}]_p), ([r_1]_{p'}, \cdots, [r_{\ell_p}]_{p'}) \leftarrow$ AJRNBS ▷ $(5k+1)\ell_p^2 + (5k+78)\ell_p\ell_{p'}$
2: $[r]_p \leftarrow ([r_1]_p, \cdots, [r_{\ell_p}]_p)$
3: $[v]_p \leftarrow [a]_p - [b]_p + [r]_p$
4: $v \leftarrow \mathrm{Reveal}([v]_p)$
5: **for** $i := 1 \cdots \ell_p$ **do**
6: **if** $v_i = 0$ **then**
7: $[c_i]_{p'} \leftarrow 1 - [r_i]_{p'}$
8: **else**
9: $[c_i]_{p'} \leftarrow [r_i]_{p'}$
10: **end if**
11: **end for**
12: $[t_B]_{p'} \leftarrow \mathrm{UnboundedFanInAnd}(([c_1]_{p'}, \cdots, [c_{\ell_p}]_{p'}))$ ▷ $5\ell_p\ell_{p'}$
13: $[t_B]_p \leftarrow \mathrm{CBBS}([t_B]_{p'})$ ▷ $(5k+1)\ell_p + (5k+2)\ell_{p'}$
14: $[t_B]_p$ を出力

A.2 AJRNBS 利用パターンの区間判定プロトコルへの適用

3.13 節で述べたように，区間判定プロトコル [9] の通信

区間判定プロトコル

Input: $[a]_p$, b, c
1: $([r_1]_p, \cdots, [r_{\ell_p}]_p) \leftarrow$ JRNBS ▷ $76\ell_p^2$
2: $[r]_p \leftarrow ([r_1]_p, \cdots, [r_{\ell_p}]_p)$
3: $[v]_p \leftarrow [a]_p + [r]_p$
4: $v \leftarrow \mathrm{Reveal}([v]_p)$
5: **if** $c \le v$ **then**
6: $d.1 = v - b$
7: $d.2 = v - c$
8: **end if**
9: **if** $v \le b$ **then**
10: $d.1 = v + p - c$
11: $d.2 = v + p - b$
12: **end if**
13: **if** $b < v < c$ **then**
14: $d.1 = v - b - 1$
15: $d.2 = v + p - c - 1$
16: **end if**
17: $[u_B.1]_p \leftarrow \mathrm{BLT}(([r_1]_p, \cdots, [r_{\ell_p}]_p), ((d.1)_1, \cdots, (d.1)_{\ell_p}))$ ▷ $17\ell_p^2$
18: $[u_B.2]_p \leftarrow \mathrm{BLT}(([r_1]_p, \cdots, [r_{\ell_p}]_p), ((d.2)_1, \cdots, (d.2)_{\ell_p}))$ ▷ $17\ell_p^2$
19: **if** $v \le b$ or $c \le v$ **then**
20: $[u_B]_p \leftarrow [u_B.1]_p \times [u_B.2]_p$ ▷ ℓ_p
21: **end if**
22: **if** $b < v < c$ **then**
23: $[u_B]_p \leftarrow 1 - [u_B.1]_p \times [u_B.2]_p$ ▷ ℓ_p
24: **end if**
25: $[u_B]_p$ を出力

量は $110\ell_p^2+\ell_p$ であり，ラウンド数は 13 である．区間判定プロトコルのステップごとの通信量を ▷ 以降に示す．通信の発生するステップを抜粋するので，全体のステップについては，3.13 節を参照いただきたい．

区間判定プロトコルのステップ 1 から 4 が，JRNBS 利用パターンにおけるステップ 1 から 4 に各々相当する．等号判定プロトコルのステップ 17 から 23 が，JRNBS 利用パターンのステップ 4（ビット演算 F）に相当する．AJRNBS 利用パターンを用いれば，区間判定プロトコルのステップ 17 から 23 を $GF(p')$ 上で実行できる．AJRNBS 利用パターンを区間判定プロトコルに対して適用した Advanced IT（AIT）プロトコルを下記に示す．上記の区間判定プロトコルにおける各ステップの通信量が，下記の AIT プロトコルにおける各ステップの通信量へと改善される一方，法変換（ステップ 25）の通信量が加わる．AIT の通信量は，下記の AIT の各ステップの通信量の合計であるため，$(5k+1)\ell_p^2+(5k+112)\ell_p\ell_{p'}+(5k+1)\ell_p+(5k+3)\ell_{p'}$ である．ラウンド数は 6.5 節で述べたように 2 増加し，15 である．

Advanced 区間判定（AIT）プロトコル

Input: $[a]_p$, b, c

1: $([r_1]_p,\cdots,[r_{\ell_p}]_p),([r_1]_{p'},\cdots,[r_{\ell_p}]_{p'}) \leftarrow$ AJRNBS ▷ $(5k+1)\ell_p^2+(5k+78)\ell_p\ell_{p'}$
2: $[r]_p \leftarrow ([r_1]_p,\cdots,[r_{\ell_p}]_p)$
3: $[v]_p \leftarrow [a]_p-[b]_p+[r]_p$
4: $v \leftarrow \mathrm{Reveal}([v]_p)$
5: **if** $c \le v$ **then**
6: $\quad d.1=v-b$
7: $\quad d.2=v-c$
8: **end if**
9: **if** $v \le b$ **then**
10: $\quad d.1=v+p-c$
11: $\quad d.2=v+p-b$
12: **end if**
13: **if** $b<v<c$ **then**
14: $\quad d.1=v-b-1$
15: $\quad d.2=v+p-c-1$
16: **end if**
17: $[u_B.1]_{p'} \leftarrow \mathrm{BLT}(([r_1]_{p'},\cdots,[r_{\ell_p}]_{p'}),((d.1)_1,\cdots,(d.1)_{\ell_p}))$ ▷ $17\ell_p\ell_{p'}$
18: $[u_B.2]_{p'} \leftarrow \mathrm{BLT}(([r_1]_{p'},\cdots,[r_{\ell_p}]_{p'}),((d.2)_1,\cdots,(d.2)_{\ell_p}))$ ▷ $17\ell_p\ell_{p'}$
19: **if** $v \le b$ or $c \le v$ **then**
20: $\quad [u_B]_{p'} \leftarrow [u_B.1]_{p'} \times [u_B.2]_{p'}$ ▷ $\ell_{p'}$
21: **end if**
22: **if** $b<v<c$ **then**
23: $\quad [u_B]_{p'} \leftarrow 1-[u_B.1]_{p'} \times [u_B.2]_{p'}$ ▷ $\ell_{p'}$
24: **end if**
25: $[u_B]_p \leftarrow \mathrm{CBBS}([u_B]_{p'})$ ▷ $(5k+1)\ell_p+(5k+2)\ell_{p'}$
26: $[u_B]_p$ を出力

p' の大きさを，6.2 節で中間秘密情報の上限を表す式 $\max(T_F,\lceil\sqrt{\ell_p}\rceil+1,n)$ を用いて考える．ビット演算 F に相当する箇所は，ステップ 17 から 23 である．ステップ 17 と 18（BLT）の秘密の中間値の上限は $\max(\lceil\sqrt{\ell_p}\rceil+1,n)$ である．ステップ 19 から 24 は 1 ビットの秘密どうしの演算であり，上限は $n=\max(1,n)$ である．したがって $T_F=\max(\lceil\sqrt{\ell_p}\rceil+1,n)$ である．以上から，p' は，$\max(\lceil\sqrt{\ell_p}\rceil+1,n)$ を超える最小の素数に設定することができる．通信量の削減効果および計算量の増加については，7.2 節で述べたとおりである．

加藤 遼 （学生会員）

2010 年電気通信大学電気通信学部人間コミュニケーション学科卒業．2012 年同大学大学院情報理工学研究科総合情報学学専攻博士前期課程修了．現在，同専攻博士後期課程在学中．情報セキュリティの研究に従事．

西出 隆志 （正会員）

1997 年東京大学理学部情報科学科卒業．同年日立ソフトエンジニアリング株式会社入社．セキュリティ，ネットワーク製品の設計開発に従事．2003 年南カリフォルニア大学修士課程修了．2008 年電気通信大学博士（工学）．2009 年九州大学大学院システム情報科学研究院助教を経て，2013 年より筑波大学システム情報系准教授．暗号，情報セキュリティの研究に従事．ACM，電子情報通信学会，各会員．

吉浦 裕 （正会員）

1981 年東京大学理学部情報科学科卒業．日立製作所を経て，2003 年より電気通信大学勤務．現在，情報理工学研究科教授．情報セキュリティ，プライバシ保護の研究に従事．博士（理学）．日立製作所社長技術賞（2000 年），情報処理学会論文賞（2005 年，2011 年），日本セキュリティ・マネジメント学会論文賞（2010 年），システム制御情報学会産業技術賞（2000 年），IEEE IIHMSP best paper award（2006）等受賞．情報処理学会，電子情報通信学会，日本セキュリティ・マネジメント学会，人工知能学会，システム制御情報学会，IEEE 各会員．