Vol.2011-CSEC-54 No.12
Vol.2011-SPT-1 No.12
2011/7/12

$GF(p)$ から $GF(q)$ への秘匿モジュロ変換プロトコルの提案

加藤 遼[†1] 桐淵直人[†1]
西脇雄高[†1] 吉浦裕[†1]

個人情報や秘密情報を保護しながら活用することがますます重要になっている．この課題を解決する手法として，秘密分散法を用いた秘匿計算が期待されている．しかし，$Shamir(k,n)$ 閾値秘密分散法のような $GF(p)$ 上の秘密分散法を用いる場合，異なる素数で分散されたシェア間で秘匿計算が行えないという問題がある．本論文では，この問題の解決する為に，ある素数で分散されたシェアを任意の別の素数でのシェアに変換する，秘匿モジュロ変換プロトコルの基本方針を提案した．基本方針に従って，$Shamir(k,n)$ 閾値秘密分散法上の秘匿モジュロ変換プロトコルを具体的に設計し，安全性とコストを評価した．また，設計した秘匿モジュロ変換プロトコルの通信量・通信回数を削減した，改良型秘匿モジュロ変換プロトコルを設計した．改良型秘匿モジュロ変換プロトコルは，積の回数は $O(lk)$，ラウンドは $O(\log k)$ となった．

Improving Efficiency of Secure Multi-Party Computation

Ryo Kato,[†1] Naoto Kiribuchi,[†1] Yutaka Nisiwaki[†1]
and Hiroshi Yoshiura[†1]

It becomes more and more important to balance information usability and confidentiality. Secure multi-party computation based on secret sharing is expected to meet this challenge. It has serious problems, however, that it requires large amount of communications and storage volume. In this paper, we describe a method to reduce the communication and storage costs of secure multi-party computation based on secret sharing.

†1 電気通信大学
The University of Electro-Communications

1. はじめに

個人の購買履歴や位置情報，診療履歴などをデータベースに集積し，直接あるいはマイニングを通じて利用する事により利用者に対して高度なサービスを提供する事が可能になる．しかし，それらの個人情報がデータベースから漏洩すれば，様々な致命的な問題が引き起こされる．またクラウドコンピューティングでは利用者の機密情報を守ると同時に，利用者のユーザービリティも高水準に保つ必要がある．したがって，情報の利用と保護を両立する事が重要となっている．

情報の利用と保護を両立する有望な技術として，秘密分散法[1)2)] に基づく秘匿計算[3)4)5)6)] が期待されている．秘匿計算では，秘密情報を n 人の参加者に分散して，参加者間の通信により秘密情報を秘匿したままの状態でその関数値を求める事が出来る．分散した秘密情報をシェアと呼ぶ．秘密分散法と秘匿計算を用いる事により，例えば個人の購買履歴を秘匿したまま，データ分析が可能となる．

しかし，$Shamir(k,n)$ 閾値秘密分散法[1)] のような $GF(p)$ 上の秘密分散法を用いる場合，異なる素数で分散されたシェア間の秘匿計算が出来ない．

この事は，秘匿計算の実用面において重大な問題を生じさせる．ここでは二つの例を挙げる．第一の例では，一日の労働時間と時給のような，値域が大きく異なる二つの属性を持った秘密分散データベースを考える．一日の労働時間と時給の積により日給を計算する為に，この二つの属性は同一の素数で分散する必要がある．そして，その素数は時給の最大値 (例えば 10000) 以上の値である．この時生じる問題は二つである．一つ目の問題は，労働時間のような 0 以上 24 以下の小さな値域の属性まで，大きな素数で分散してしまう事から生じる，データ容量の増大ならびに通信量の増大である．二つ目の問題は，ボーナスのような，現在の素数の範囲域に収まらない新たな属性をこのデータベースに加えたい時，現状ではシェアを一度復元してから，ボーナスの最大値以上の素数で再度分散し直す必要がある事から生じる，利便性および安全性の低下である．第二の例では，組織 A と組織 B が存在し，組織 A はデータ A' を，組織 B はデータ B' を秘密分散して安全に管理している状況を考える．この時，組織 A と B は異なるポリシーに基づくので，データ A' と B' は同じ素数で分散されているとは限らない．もしも同じ素数ではない場合，現状では，一方のデータの中身を一端復元し，他方で使われている素数で分散し直す必要がある．しかし，この方法は，利便性および安全性の低下が生じる．

本研究では，この問題を解決する為に秘匿モジュロ変換を提案した．秘匿モジュロ変換は，

Vol.2011-CSEC-54 No.12
Vol.2011-SPT-1 No.12
2011/7/12

素数 p で分散されたシェアを異なる任意の素数 q でのシェアに変換する秘匿計算である．この変換の過程で, 秘密情報は一度も復元されない．秘匿モジュロ変換を用いれば, 素数 p でのシェアと素数 p' でのシェアの間の秘匿計算は, 両方のシェアを素数 q に統一する事で, 素数 q でのシェア同士の秘匿計算として実行可能である．

この秘匿モジュロ変換の具体例を, $Shamir(k,n)$ 閾値秘密分散法上での秘匿計算で, 二種類, 設計した．基礎型秘匿モジュロ変換と, 改良型秘匿モジュロ変換である．基礎型秘匿モジュロ変換は積の回数は $O(l\log l+lk)$, ラウンドは $O(\log k)$, 改良型秘匿モジュロ変換は積の回数は $O(lk)$, ラウンドは $O(\log k)$ となった．

2. 先 行 研 究

2.1 秘密分散法

2.1.1 $Shamir(k,n)$ 閾値秘密分散法

分散の際は, ディーラは先ず秘密情報 $S\in Z_p$ となる $GF(p)$ 上の $k-1$ 次の多項式

$$f(x)=S+r_1x+\cdots+r_{k-1}x^{k-1}\pmod p$$

を選ぶ．ただし $S<p$ かつ, $r_i\in Z_p$ は乱数である．次に参加者 P_d に対し, ディーラは $f(d)$ を配布する $(1\leq d\leq n)$.

復元の際は, k 個以上の $f(d)$ から S が得られる[1)]．

以降, $f(d)$ をシェアと呼ぶ．

2.1.2 排他的論理和を用いた秘密分散法

排他的論理和を用いた秘密分散法では, 分散の際に参加者 P_d に対し, ディーラは x_d を配布する $(1\leq d\leq n)$.

$$S=x_1\oplus x_2\oplus\cdots\oplus x_n$$

なお, S は秘密情報である．以降, x_d をシェアと呼ぶ．

2.1.3 $Shamir(k,n)$ 閾値秘密分散法と排他的論理和を用いた秘密分散法の相違点

排他的論理和を用いた秘密分散法と, $Shamir(k,n)$ 閾値秘密分散法の相違点は, 二つある．第一に, $Shamir(k,n)$ 閾値秘密分散法では秘密情報の復元が n 個のシェアの内の k 個が集まれば可能である一方, 排他的論理和を用いた秘密分散法では n 個すべてのシェアが必要である．第二に, $Shamir(k,n)$ 閾値秘密分散法では秘密情報の取り得る値域が $GF(p)$ の要素である一方, 排他的論理和を用いた秘密分散法では秘密情報は有限体ではなく, 任意の整数である．本研究では, 第二の相違点を利用する．

2.2 秘 匿 計 算

2.2.1 $Shamir(k,n)$ 閾値秘密分散法を用いた秘匿計算

$GF(p)$ 上で分散された秘密情報同士の, 和・積・大小判定・等号判定等が可能である[3)4)5)]．しかし, その一方で, $GF(p)$ 上で分散された秘密情報と $GF(q)$ 上で分散された秘密情報の間での, 秘匿計算の手法は示されていない．現時点では, 同じ素数を用いた時のみの秘匿計算が可能である．

2.2.2 $Shamir(k,n)$ 閾値秘密分散法を用いた秘匿計算の問題点

異なる素数を用いたシェア間での秘匿計算が出来ない事で生じる問題点を, 二つの状況を仮定して説明する．

状況 1　例えば, 一日の労働時間 $(0\leq workhour\leq 24)$ のような 5 ビット以内の値域を持つ属性と, 時給 $(0\leq paymentbyhour\leq 10000)$ のような 14 ビット以内の値域を持つ属性の両方を持ったデータベースが存在すると仮定する．このデータベースを分散するには, 二つの方法がある．第一が, 一日の労働時間を 5 ビットの素数 31 で分散し, 時給を 14 ビットの素数 10007 で分散する方法である．第二が両方の属性を, 素数 10007 で分散する手法である．しかし, 第一の方法は, 一日の労働時間のシェアと時給のシェアから, 日給のシェアを計算する事が出来ない．一方, 第二の方法を取ったとすると, 三つの問題が生じる．

問題点 1.1　データベースが保管する, データの容量が増大する．10007 で分散された一日の労働時間のシェアは, 0 以上 10007 未満の値域を取る．したがって, 分散される以前は 5 ビットのデータであり, 31 で分散されていれば 5 ビットのままであったデータが, 10007 で分散された時は 14 ビットのデータとなり, 3 倍のデータ量となる．

問題点 1.2　大小判定や等号判定等を含む殆どの秘匿計算の通信量は, 秘密情報を分散する素数のビット数に比例する．したがって, 労働時間が 10 時間以上の人間を捜すような秘匿計算を行う時, 一日の労働時間が 10007 で分散されてる時と, 31 で分散されている時で比べれば, 前者の方が後者に比べ, 3 倍の通信量を要する．

問題点 1.3　このデータベースに新たに, 10007 以上のボーナスの属性 $(0\leq bonus\leq$

Vol.2011-CSEC-54 No.12
Vol.2011-SPT-1 No.12
2011/7/12

1000000) を追加したい場合, 現状では一度, データベース内のシェアを一度, 秘密情報に復元し, 10007 以上の素数で新たに分散させる必要がある.

状況 2　二つの秘密分散データベースが存在している. それぞれのデータベースを A, B と名付ける. データベース A 内部の秘密情報は素数 p で分散され, データベース B 内部の秘密情報は素数 p' で分散されている. このような状況は, 異なる組織間のデータベースでは一般的に生じる.

問題点 2.1　データベース A と B を統合して, 新たなデータベース C を作りたい. なお, データベース C 内部のシェアは, q で分散されているとする ($p \leq q, p' \leq q$).

現状では, このような統合を可能にするには, データベース A, B 内部のシェアを一度, 秘密情報に復元し, 復元した秘密情報を再度素数 q で分散する, という手法を取らざるを得ない. しかし, 秘密情報を一度復元するこの手法は, 利便性と安全性の低下の面から好ましいとはいえない

3. 表記規則

3.1 シェア

シェアおよび, その他の記号について, 以降, 次のように表記する.

- p, q：
  素数
- l：
  素数 p のビット数
- S：
  秘密情報 S(ただし, S は $GF(p)$ 上)
- $S_l, S_{l-1}, \ldots, S_1$：
  秘密情報 S のビット
- $[S]_p$：
  $Shamir(k, n)$ 閾値秘密分散法のような, $GF(p)$ 上で分散された秘密情報 S のシェア
- $[S]$：
  排他的論理和を用いた秘密分散法のような, 整数上で分散された秘密情報 S のシェア

3.2 $GF(p)$ 上の秘匿計算

$GF(p)$ 上の秘匿計算について, 以後, 次のように表記する.

- $a \leftarrow Reveal([a]_p)$：
  全参加者に a を公開する秘匿計算
- $[a_l]_p, [a_{l-1}]_p, \ldots, [a_1]_p \leftarrow BitDecomposition([a]_p)$：
  $[a]_p$ をビットのシェアへ分解する秘匿計算
- $[a+b]_p \leftarrow [a]_p + [b]_p$：
  $[a]_p$ と $[b]_p$ から, $[a+b]_p$ を求める秘匿計算
- $[a \times b]_p \leftarrow [a]_p \times [b]_p$：
  $[a]_p$ と $[b]_p$ から, $[a \times b]_p$ を求める秘匿計算
- $[a > b]_p \leftarrow [a]_p > [b]_p$：
  $[a]_p$ と $[b]_p$ から, $[a > b]_p$ を求める秘匿計算
- $[a > b]_p \leftarrow [a_l]_p, [a_{l-1}]_p, \ldots, [a_1]_p > b$：
  $[a_1]_p, [a_2]_p, \ldots, [a_l]_p$ と b から, $[a > b]_p$ を求める秘匿計算
- $[a \oplus b]_p \leftarrow [a]_p \oplus [b]_p$：
  $[a]_p$ と $[b]_p$ から, $[a \oplus b]_p$ を求める秘匿計算

4. 秘匿モジュロ変換

4.1 達成目標

$[a]_p$ と $[b]_q$ から, $[f(a,b)]_q$ を求める秘匿計算を, 本研究の目標とする.

$[a]_q$ と $[b]_q$ から, $[f(a,b)]_q$ を求める秘匿計算が存在する時に, この目標を達成する為には, $[a]_p$ から $[a]_q$ を求める秘匿計算が存在すればよい. 本研究では, $[a]_p$ から $[a]_q$ を求める秘匿計算を, 秘匿モジュロ変換と呼称し, その具体的手法を提案する.

4.2 適用範囲

$Shamir(k, n)$ 閾値秘密分散法のような, $GF(p)$ 上の秘密分散法に基づく和・積が可能な秘匿計算を対象として, 次の基本方針を提案する.

4.3 基本方針

秘匿モジュロ変換は次の $Step$ により構成される (図 1).

Vol.2011-CSEC-54 No.12
Vol.2011-SPT-1 No.12
2011/7/12

Step	概要
	$[S]_p$
Step1	↓
	$[S_l]_p, [S_{l-1}]_p, \ldots, [S_1]_p$
Step2	↓
	$[S_l], [S_{l-1}], \ldots, [S_1]$
Step3	↓
	$[S_l]_q, [S_{l-1}]_q, \ldots, [S_1]_q$
Step4	↓
	$[S]_q$

図 1 秘匿モジュロ変換の基本方針

- Step1　$GF(p)$ 上のシェアを, $GF(p)$ 上のビットのシェアに分解する.
- Step2　$GF(p)$ 上のビットのシェアを, 整数上のビットのシェアに変換する.
- Step3　整数上のビットのシェアを, $GF(q)$ 上のビットのシェアに変換する.
- Step4　$GF(q)$ 上のビットのシェアを, $GF(q)$ 上のシェアに変換する.

5. 基礎的な秘匿モジュロ変換

上記の基本方針にのっとり, $Shamir(k, n)$ 閾値秘密分散上での秘匿モジュロ変換を提案する.

5.1 手　　法

Step1　秘匿計算で, $[S]_p$ をビットのシェアへ分解する.

$$[S_l]_p, [S_{l-1}]_p, \ldots, [S_1]_p \leftarrow BitDecomposition([S]_p)$$

Step2.1　1 人目から $k-1$ 人目までの参加者の内, 参加者^i は l ビットの乱数 R^i を生成する. R^i のビットのシェア $[R^i_1]_p, [R^i_2]_p, \ldots [R^i_l]_p$ を全参加者に分散する $(1 \leq i \leq k-1)$.

Step2.2　k 人目の参加者の為に, 秘匿計算で, $[R^k_j]_p$ を計算する $(1 \leq j \leq l)$.

$$[R^k_j]_p \leftarrow [S_j]_p \oplus [R^1_j]_p \oplus [R^2_j]_p \oplus \cdots \oplus [R^{k-1}_j]$$

Step2.3　参加者^k は R^k_j を復元する.

Step3.1　参加者^i は $[R^i_l]_q, [R^i_{l-1}]_q, \ldots, [R^i_1]_q$ を全参加者に分散する $(1 \leq i \leq k)$.

Step3.2　秘匿計算で, $[S_j]_q$ を計算する $(1 \leq j \leq l)$.

$$[S_j]_q \leftarrow [R^1_j]_q \oplus [R^2_j]_q \oplus \cdots \oplus [R^k_j]_q$$

Step4　秘匿計算で, $[S]_q$ を計算する.

$$[S]_q \leftarrow 2^0[S_1]_q + 2^1[S_2]_q + \cdots + 2^{l-1}[S_l]_q$$

5.2 具　体　例

具体例を次に示す. $Shamir(2, 3)$ 閾値秘密分散法として, $S = 5$, $p = 11$, $l = 4$, $q = 17$ とする.

Step1　秘匿計算で, $[5]_{11}$ をビットのシェアへ分解する.

$$[0]_{11}, [1]_{11}, [0]_{11}, [1]_{11} \leftarrow BitDecomposition([5]_{11})$$

Step2.1　参加者^1 が $R^1 = 9 = 2^0 + 2^3$ を生成する. $[1]_{11}, [0]_{11}, [0]_{11}, [1]_{11}$ を全参加者に分散する.

Step2.2　2 人目の参加者の為に, 秘匿計算で, $[R^2_j]_{11}$ を計算する.

$$[R^2_j]_{11} \leftarrow [S_j]_{11} \oplus [R^1_j]_{11}$$
$$[1]_{11} \leftarrow [0]_{11} \oplus [1]_{11}$$
$$[1]_{11} \leftarrow [1]_{11} \oplus [0]_{11}$$
$$[0]_{11} \leftarrow [0]_{11} \oplus [0]_{11}$$

Vol.2011-CSEC-54 No.12
Vol.2011-SPT-1 No.12
2011/7/12

$[0]_{11} \leftarrow [1]_{11} \oplus [1]_{11}$

Step2.3 参加者^2 は $12 = 2^2 + 2^3$ を復元する.

Step3.1 参加者^1 が $[1]_{17}, [0]_{17}, [0]_{17}, [1]_{17}$ 、参加者^2 が $[1]_{17}, [1]_{17}, [0]_{17}, [0]_{17}$ を分散する.

Step3.2　秘匿計算で, $[S_j]_{17}$ を計算する.

$[S_j]_{17} \leftarrow [R_j^1]_{17} \oplus [R_j^2]_{17}$
$[0]_{17} \leftarrow [1]_{17} \oplus [1]_{17}$
$[1]_{17} \leftarrow [0]_{17} \oplus [1]_{17}$
$[0]_{17} \leftarrow [0]_{17} \oplus [0]_{17}$
$[1]_{17} \leftarrow [1]_{17} \oplus [0]_{17}$

Step4　秘匿計算で, $[S]_{17}$ を計算する.

$[5]_{17} \leftarrow 2^0[1]_{17} + 2^1[0]_{17} + 2^2[1]_{17} + 2^3[0]_{17}$

5.3 安　全　性

基礎的な秘匿モジュロ変換の安全性を示す.

*Step*2.3 における秘密情報の復元を除くすべての *Step* は秘匿計算で行われているため, その安全性は秘匿計算の安全性に帰着される. したがって, *Step*2.3 で安全性が低下していない事を示せれば, 基礎的な秘匿モジュロ変換の安全性は秘匿計算の安全性に帰着される.

*Step*2.3 においては, 参加者^k だけに R_j^k が復元されている. R_j^k はその実, 排他的論理和を用いた秘密分散法における S_j のシェアに等しい. S_j のシェアが k 個集まらない限り, S_j は安全である. *Step*2.3 終了時点で, 参加者^i が知るのは R_j^i のみで, 他の参加者の持つ $[S_j]$ は知らない状況が作られる. したがって, 基礎的な秘匿モジュロ変換の安全性は $Shamir(k, n)$ 閾値秘密分散法の安全性と同等である.

5.4 通信量・通信回数

上記の秘匿計算は, 積の回数は $O(l \log l + lk)$ となり, ラウンドは $O(\log k)$ となる.

分散を積の秘匿計算の 1 回分と換算する. 積の秘匿計算はそれぞれの参加者が自分以外の $n-1$ 人の参加者にシェアを送信するという点で, $n(n-1)$ の通信量を要する. これは分散における通信量と等しい.

参加者一人への秘密情報の公開を積の秘匿計算の $1/n$ 回分とする. $n-1$ 人の参加者がシェアを一人の参加者に送信するという点で, $n-1$ の通信量を要するためである.

排他的論理和を積の秘匿計算の 1 回分とする. 排他的論理和の秘匿計算内で, 積プロトコルが 1 回だけ用いられるからである.

*Step*2.2 および *Step*3.2 では, それぞれ, 積の回数は lk, ラウンドは $\log k$ を必要とする. したがって, *Step*2 から *Step*4 まででは, 積の回数は $2lk+2+1/n$, ラウンドは $2\log k+2+1/n$ である.

*Step*1 で用いる $BitDecompositionProtocol$ は積の回数が $31l \log l + 71l + 30\sqrt{l}$, ラウンドが 23 である[6]. したがって全ての *Step* の合計は, 積の回数は $31l \log l + 71l + 30\sqrt{l} + 2lk + 2 + 1/n$, ラウンドは $2\log k + 25 + 1/n$ となる.

5.5 表 記 規 則

以降, *Step*2 から *Step*3 までの $[S_l]_p, [S_{l-1}]_p, \ldots, [S_1]_p$ から $[S_l]_q, [S_{l-1}]_q, \ldots, [S_1]_q$ へ変換する秘匿計算を以後, 次のように表記する.

$[S_l]_q, [S_{l-1}]_q, \ldots, [S_1]_q \leftarrow Convert([S_l]_p, [S_{l-1}]_p, \ldots, [S_1]_p)$

この秘匿計算における積の回数は $O(lk)$, ラウンドは $O(\log k)$ となる.

6. 改良型秘匿モジュロ変換の提案

基礎的なモジュロ変換では, $BitDecomposition$ プロトコルがその秘匿計算の通信回数・通信量の大半を占めている事が判る. したがって, $BitDecomposition$ プロトコルの使用は好ましくない.

改良手法として, $BitDecomposition$ プロトコルを用いない形での秘匿モジュロ変換プロトコルを提案する.

6.1 手　　法

Step1　秘匿計算で $GF(p)$ 上のビットの乱数のシェア $[R_l]_p, [R_{l-1}]_p, \ldots, [R_1]_p$ を得る $(R_i \in \{0, 1\})$.

Step2　秘匿計算で $GF(p)$ 上のビットの乱数のシェアを $GF(q)$ 上のビットの乱数のシェアに変換する.

$[R_l]_q, [R_{l-1}]_q, \ldots, [R_1]_q \leftarrow Convert([R_l]_p, [R_{l-1}]_p, \ldots, [R_1]_p)$

Step3　秘匿計算で $GF(p)$ 上のビットの乱数のシェア, $GF(q)$ 上のビットの乱数のシェアそれぞれから, $[R]_p, [R]_q$ を得る.

Vol.2011-CSEC-54 No.12
Vol.2011-SPT-1 No.12
2011/7/12

$$[R]_p \leftarrow 2^0[R_1]_p + 2^1[R_2]_p + \cdots + 2^{l-1}[R_l]_p$$

$$[R]_q \leftarrow 2^0[R_1]_q + 2^1[R_2]_q + \cdots + 2^{l-1}[R_l]_q$$

Step4　秘匿計算で S と R の和のシェアを得て, 公開する.

$$[S+R]_p \leftarrow [S]_p + [R]_p$$
$$S + R \leftarrow Reveal([S+R]_p)$$

Step5　秘匿計算で, S と R の和が p 以上かどうか, 判定する. $S+R>p$ の判定結果は $R > S+R$ の判定結果と等しい.

$$[R > S+R]_p \leftarrow [R_l]_p, [R_{l-1}]_p, \ldots, [R_1]_p > S+R$$

Step6　$R > S+R$ の判定の結果のシェアを $GF(q)$ 上のシェアに変換する.

$$[R > S+R]_q \leftarrow Convert([R > S+R]_p)$$

Step7　以下の秘匿計算で, $[S]_q$ を得る.

$$[S+R]_q \leftarrow S + R + p \times [R > S+R]_q$$
$$[S]_q \leftarrow [S+R]_q - [R]_q$$

6.2 原　　理

改良型秘匿モジュロ変換が正しい事を以下に示す.

*Step*7 において, $[S+R]_q$ を求める秘匿計算は, 次のように, $R > S+R$ が真の時も, 偽の時も両方において成り立つ (表 1). したがって, 正しい動作が保証される.

表 1　Step7 の正当性

	$S+R>p$	$S+R \not> p$
$[R > S+R]_q$	$[1]_q$	$[0]_q$
$S+R \pmod p$	$S+R-p$	$S+R$
$[S+R]_q$	$S+R-p+[p]_q$	$S+R+[0]_q$

6.3 安　全　性

Convert 秘匿計算の安全性は, 既に説明されている.

*Step*4 では, S に乱数を加えたものが公開されているため, 公開された値からの S の逆算は不可能である.

Convert 秘匿計算および *Step*4 を除く, すべての *Step* は秘匿計算で行われているため, その安全性は秘匿計算の安全性に帰着される.

6.4 通信量・通信回数

改良型秘匿モジュロ変換は積の回数は $O(lk)$, ラウンドは $O(\log k)$ となる.

*Step*1 のビットの乱数の秘匿計算が, 積の回数は $76l$, ラウンドは 7 となる[3].

*Step*2 および, *Step*4 の *Convert* 秘匿計算がそれぞれ, 積の回数が $2lk+2+1/n$, ラウンドは $2\log k + 2 + 1/n$ となる.

*Step*5 で, $BitwiseLess-ThanProtocol$ が積の回数は $17l$, ラウンドは 7 となる[3].

k と l では l の方が大きくなると考えられる為, *Step*2 の *Convert* 秘匿計算よりも, *Step*5 の $BitwiseLess-ThanProtocol$ の方がラウンドが大きくなると考えられる. したがって, すべての *Step* の合計は積の回数は $93l+4lk+4+2/n$, ラウンドは $16+2\log k+1/n$ となる.

7. 秘匿モジュロ変換の汎用性

秘匿モジュロ変換は $GF(p)$ 上の秘密分散法に基づく和・積が可能な秘匿計算であれば, 基本方針にしたがって設計可能である.

秘匿モジュロ変換の基本方針は $BitDecomposition$ および排他的論理和の秘匿計算からなる. そして $BitDecomposition$ も排他的論理和も和・積の秘匿計算の組み合わせからなる[3][6].

8. ま　と　め

$Shamir(k,n)$ 閾値秘密分散法のような $GF(p)$ 上の秘密分散法を用いる場合, 異なる素数で分散されたシェア間で秘匿計算が行えないという問題があった. この問題の解決には, 素数 p で分散されたシェアを, 任意の素数 q で分散されたシェアに変換する秘匿計算法が存在すればよい. 本研究ではこの秘匿計算法として, 秘匿モジュロ変換を提案した.

本研究で, $GF(p)$ 上の秘密分散法に基づく和・積が可能な秘匿計算を対象として, 秘匿モ

Vol.2011-CSEC-54 No.12
Vol.2011-SPT-1 No.12
2011/7/12

ジュロ変換の基本方針を示した．その基本方針にのっとり，$Shamir(k,n)$ 閾値秘密分散法上の秘匿計算を対象として，具体的な秘匿モジュロ変換を設計した．この秘匿モジュロ変換は，積の回数は $O(l\log l+lk)$ となり，ラウンドは $O(\log k)$ となった．

さらに，このプロトコルの通信量およびラウンドの大半を，$BitDecomposition$ プロトコルが占めているとの知見を得て，$BitDecomposition$ プロトコルを用いない改良型秘匿モジュロ変換を設計した．改良型秘匿モジュロ変換は積の回数は $O(lk)$，ラウンドは $O(\log k)$ となった．

参 考 文 献

1) A. Shamir, “How to Share a Secret,” Commun. ACM, vol.22, No.11, pp612–613, 1979.
2) R. Gennaro, M. O. Rabin and T. Rabin, “Simplified VSS and fast-track multiparty computations with applications to threshold cryptography,” Proc. of the 17th annual ACM symposium on Principles of distributed computing, pp101–111, 1998.
3) T. Nishide and K. Ohta, “Multiparty Computation for Interval, Equality, and Comparison Without Bit-Decomposition Protocol,” Proc. 10th International Conference on Theory and Practice of Public-Key Cryptography(PKC), LNCS 4450, pp343–360, 2007.
4) SecureSCM, “Security Analysis,” Technical Report D9.2, SecureSCM, 2009.
5) Damgård, M. Fitzi, E. Kiltz, J. B. Nielsen and T. Toft, “Unconditionally Secure Constant-Rounds Multi-party Computation for Equality, Comparison, Bits and Exponentiation,” Proc. 3rd Theory of Cryptography Conference (TCC 2007), LNCS 3876, pp285–304, 2006.
6) T. Toft, “Constant-Rounds, Almost-Linear Bit-Decomposition of Secret Shared Values,”
Proc. CT-RSA 2009, LNCS 5473, pp357–371, 2009.
7) J. Bar-Ilan and D. Beaver, “Non-cryptographic fault-tolerant computing in constant number of rounds of interaction,” Proc. of the 8th annual ACM Symposium on Principles of distributed computing, pp201–209, 1989.
8) M. Ben-Or, S. Goldwasser and A. Wigderson, “Completeness theorems for non-cryptographic fault-tolerant distributed computation,” Proc. of the 20th annual ACM Symposium on Theory of computing, pp1–10, 1988.
9) R. Cramer and I. Damgård, “Secure Distributed Linear Algebra in a Constant Number of Rounds,” Proc. Crypto 2001, LNCS 2139, pp119–136, 2001.
10) R. Cramer, I. Damgård, and Y. Ishai, “Share conversion, pseudorandom secret sharing and applications to secure computation,” Proc. 2nd Theory of Cryptography Conference, LNCS 3378, pp342–362, 2005.
11) C. Ning and Q. Xu, “Multiparty Computation for Modulo Reduction without Bit-Decomposition and A Generalization to Bit-Decomposition,” Proc. AsiaCrypt 2010, LNCS 6477, pp483–500, 2010.