情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2010-CSEC-50 No.45
2010/7/2

マルチパーティ計算による効率的なビット分解プロトコル

千田浩司[†1]　五十嵐大[†1]　高橋克巳[†1]

ビット分解プロトコルは，マルチパーティ計算の計算量や通信量を削減できる有用なツールとして知られている．ただしビット分解プロトコル自体がマルチパーティ計算のボトルネックとなる場合があり，当該プロトコルの処理効率向上が求められる．本稿では，特に通信量に優れるビット分解プロトコルを提案する．また，従来検討されてきた素体上や乗法群上だけでなく環 $\mathbb{Z}/2^{\ell}\mathbb{Z}$ 上の演算も可能となるため，剰余演算を効率良く処理する効果も見込める．

An Efficient Multiparty Bit-decomposition Protocol

Koji Chida,[†1] Dai Ikarashi[†1]
and Katsumi Takahashi[†1]

The bit-decomposition protocol is known as a useful tool that can reduce the computation and communication costs of the multiparty computation. However, the bit-decomposition protocol might become the bottleneck of the multiparty computation thereby be required to be more efficient. In this paper we propose a traffic-efficient multiparty bit-decomposition protocol. In addition, the underlying modular operations of our protocol can be done faster than those of the existing most bit-decomposition protocols because our protocol works over ring $\mathbb{Z}/2^{\ell}\mathbb{Z}$ unlike the previous ones.

1. はじめに

情報を安全に活用し，情報漏洩やプライバシ侵害から守ることは，情報セキュリティ分野における長年の課題である．この課題に対する一つのアプローチとして，入力データを秘匿しつつ各種計算を可能とする**秘匿関数計算** (Secure Function Evaluation)[1] が古くから知られている．秘匿関数計算は，実用に向けた課題として，所望の計算の複雑さに応じて計算量や通信量が膨大となることが挙げられるが，鍵を秘匿しつつ AES (Advanced Encryption Standard) の暗号化・復号を実装した例[2] 等，最近では実用を見据えた動きが各所で見られる．

本研究の目的は，秘匿関数計算を用いて，医療情報や生活行動情報等の個人のプライバシに関わる情報を安全に利用することであり，特に統計情報を得るための活用を想定している．したがって大規模データ処理が不可欠となり，秘匿関数計算のより一層の処理効率向上が技術課題である．

秘匿関数計算は一般に，所望の計算の入力値を秘匿処理したうえで提供する主体と，その入力値を復元すること無く処理する主体の少なくとも 2 主体が関与し，秘匿方法は暗号化や秘密分散が代表的である．Yao は，所望の計算を論理回路で記述し，複数主体の協調計算により秘匿関数計算を実現する**マルチパーティプロトコル**または**マルチパーティ計算**と呼ばれる手法を提案した[3]．現在まで，論理回路演算を実行するマルチパーティ計算として，紛失通信（Oblivious Transfer）を利用した方式[4] や，MIX-net を利用した方式[5] 等が提案されている．なお論理回路演算の実行を可能とする秘匿関数計算は**秘匿回路計算** (Secure Circuit Evaluation) と呼ばれる．

一方，環 $\mathbb{Z}/m\mathbb{Z}$（m は適当な整数）上または乗法群 $(\mathbb{Z}/m\mathbb{Z})^{\times}$ 上の算術演算を可能とする秘匿関数計算も提案されており，秘密分散に基づく方式[6),7] や準同型暗号に基づく方式[8),9] が知られている．これにより，加算や乗算においては秘匿回路計算よりも十分な高速処理が期待できる．また，秘匿回路計算の欠点である処理効率の大幅な悪化を抑止するため，秘匿関数計算を論理回路演算と算術演算に分け双方を実行可能とし，全体の処理効率を上げる**ビット分解プロトコル** (Bit-Decomposition Protocol) もいくつか提案されている[10)–13)]．ビット分解プロトコルは，秘匿された整数を入力として，当該整数に関する情報を漏らすこと無く当該整数のビットごとの秘匿された値を得る手法である．

前述した既存のマルチパーティ計算[4)–9)] は何れも，加減算および定数倍は各主体が単独で効率良く実行できる．また復元処理は乗算と比べると少ない計算量，通信量で済む．したがって，乗算の秘匿関数計算の回数を抑えることが全体の処理効率の向上につながることが期待できる．

本稿では，乗算の秘匿関数計算を不要とできるビット分解プロトコルを提案する．また m の制限が無いため，例えば $m = 2^{\ell}$ (ℓ は適当な整数) とすることで，剰余演算が上位ビッ

†1 NTT 情報流通プラットフォーム研究所
NTT Information Sharing Platform Laboratories

© 2010 Information Processing Society of Japan

トの切り捨てで済み，$\mathbb{Z}/2^\ell\mathbb{Z}$ 上で動作する秘匿関数計算[⋆1]に適用することで効率良く処理できるようになる．従来は m が (大きな) 素数，あるいは大きな素数の積であることを前提としており，$m = 2^\ell$ とした場合のマルチパーティ計算やビット分解プロトコルについては筆者らが知る限り議論されていない．

2. 準　　備

本稿では整数 a を秘匿した値に関して以下の表記を用いる．

- $[a]_m$: ある整数 m (≥ 2) について $a \bmod m$ を秘匿した値
- $[a]_B = ([a_{\ell-1}]_m, \ldots, [a_0]_m)$ ただし $a = \sum_{i=0}^{\ell-1} 2^i a_i$ ($a_i \in \{0,1\}$), $\ell = \lceil \log_2 m \rceil$

すると秘匿関数計算は $a, a', c \in \mathbb{Z}$ および $b, b' \in \{0,1\}$ に関する有意な情報を漏らすこと無く以下の何れかを実現する．

- 算術演算: $[a]_m, [a']_m, c \to [a \pm a']_m, [a \pm c]_m, [aa']_m, [ac]_m, [a^{-1}]_m$
- 論理演算: $[b]_m, [b']_m \to [bb']_m, [1-b]_m$

ただし $[a^{-1}]_m$ は $\mathrm{GCD}(m, a) = 1$ が必要十分である．

上記より明らかに，算術演算の秘匿関数計算は論理演算も実現できる．しかし $a \notin \{0,1\}$ であれば $[a]_m \to [a]_B$，すなわちビット分解プロトコルが必要となる．

3. 関 連 研 究

Damgård らは定数回の通信で済むビット分解プロトコルを提案し[11]，西出らは更に通信回数，通信量，および乗算の秘匿関数計算の回数を削減する改良方式を提案した[13]．何れの方式も乗算の秘匿関数計算を $O(\ell \log \ell)$ 回必要とする．Schoenmakers らは，文献 8) のマルチパーティ計算を対象として，乗算の秘匿関数計算の回数および通信量をそれぞれ $O(\ell)$, $O(n\ell^2)$ とできるビット分解プロトコルを提案した[12]．ここで n は秘匿関数計算に参加する主体の数とする．ただし通信回数が $O(\ell)$ となり，トレードオフの関係を持つ別の手法も合わせて提案している．文献 12) では，それぞれ LSB Gate, LSBs Gate, BITREP Gate と呼ばれる三つの異なる手法が提案されている．

上述の既存方式は，LSBs Gate および BITREP Gate 以外は逆元の計算を必要とする．そして LSBs Gate および BITREP Gate においても，通信回数を定数とするためには逆元の計算を必要とする．特に偶数の逆元を避ける方法は自明でないため，$m = 2^\ell$ とした場合は直接適用できない．以下では，LSB Gate および LSBs Gate の概略を通じて逆元の要否をみる[⋆2]．

LSB Gate は先ず $[a]_m$ から $[a_0]_m$ を求める．すなわち a の最下位ビットを秘匿した値を求める．次に $[(a-a_0)/2]_m$ を計算し，$(a-a_0)/2$ の最下位ビット，すなわち a の下位 2 番目のビットを秘匿した値を求める．以下同様の処理を繰り返し $[a]_B$ を得る．したがって 2 の逆元の存在が前提となるため，少なくとも m を奇数とする必要がある．

一方 LSBs Gate は先ず乱数を秘匿した値 $[r]_m$ および $[r]_B$ を求める．ここで r はどの主体も知り得ないように生成する．文献 12) では 3 通りの解法が示されているが，そのうちの一つである定数回通信を実現する解法は，平文に依存した値の逆元の計算が必要となり，m の取り方が制限されてしまう．次に $[a-r]_m$ を計算して $d = a - r$ を復元する．ここで d は負の値となる場合があるが，復元した結果は $d = a - r \bmod m$，すなわち 0 以上 m 未満の整数となる．そのため d の符号を別途判定する必要がある．最終的に，加算回路の秘匿関数計算を実行することで d および $[r]_B$ から $[d+r]_B = [a]_B$ を得る．例えば以下の加算回路の計算式より $[a_i]_m$ ($i = 0, \ldots, \ell-1$) を求めることができる．

$$
\begin{aligned}
e_i &= d_i + r_i - 2d_i r_i \\
a_i &= e_i + c_i - 2e_i c_i \ (c_0 = 0) \\
c_{i+1} &= d_i r_i + e_i c_i - d_i r_i e_i c_i
\end{aligned}
$$

上記計算式を秘匿関数計算により実行する場合，乗算の秘匿関数計算が $2\ell - 1$ 回必要となる．c_{i+1} は再帰的に計算され並列化できないため，通信回数は $O(\ell)$ となる．

4. 提 案 方 式

LSB Gate は，$a - \sum_{i=0}^{h-1} 2^i a_i$ の下位 h ビットが全て 0 となることを利用し，$a - \sum_{i=0}^{h-1} 2^i a_i$ に 2^h の逆元を乗じることで，h ビット右シフト処理を秘匿関数計算により実現している．そしてこれにより，下位から順に a のビットを秘匿した値を求めている．提案方式は，このシフト処理を行わないまま処理を継続することで逆元を不要とする．これにより m の制限を無くすことができるとともに，乗算の秘匿関数計算を不要とできる．

具体的な処理を以下に示す．

(1)　$[a]_m$ を入力する．

⋆1 例えば筆者らが提案した手法がある[14]．

⋆2 理解の容易さを優先し，一部正確でない記述がある．正しくは文献 12) を参照されたい．なお BITREP Gate は，LSBs Gate における平文空間の制限を無くしているが，LSBs Gate と比べて処理効率は優位とならない．

© 2010 Information Processing Society of Japan

(2) $h = 0$ から $\ell - 1$ について以下を行う．

(3) 乱数 $r \in_R \{0, \ldots, m-1\}$ を秘匿した値 $[r]_m$ および $[r_h]_m$ を求める．ここで LSBs Gate 同様，r はどの主体も知り得ないように生成する．

(4) $[a]_m$, $\{[a_i]_m\}_{i=0}^{h-1}$, および $[r]_m$ から $[a - \sum_{i=0}^{h-1} 2^i a_i - r]_m$ を計算する．

(5) $d = a - \sum_{i=0}^{h-1} 2^i a_i - r \bmod m$ を復元する．

(6) $c_0 = 0$ とし，$j = 0$ から $h-1$ まで，$c_{j+1} = (r_j \wedge d_j) \vee ((r_j \oplus d_j) \wedge c_j)$ を計算する．ここで $(a - \sum_{i=0}^{h-1} 2^i a_i)_j = r_j \oplus d_j \oplus c_j = 0$ が成り立つことから (r_j, d_j はそれぞれ r, d の下位 $j+1$ 番目のビット)，各主体は平文の処理により c_{j+1} を単独で求めることができる．

(7) $[r_h]_m$ および d_h から，

$$[a_h]_m = \begin{cases} [r_h + d_h - 2r_h d_h]_m & (c_h = 0) \\ [1 - (r_h + d_h - 2r_h d_h)]_m & (c_h = 1) \end{cases}$$

を計算して出力する．

5. 評　　価

提案方式は乱数を秘匿した値を求める必要があるが，これは既存方式[8),11),13)] も同様である．以下では乱数を秘匿した値の計算は事前処理されるものとして評価に含めないものとする．

提案方式は乗算の秘匿関数計算を必要としないことが，他方式と異なる最も大きな特徴である．しかし定数倍や加減算の秘匿関数計算の実行回数が他方式よりも悪く $O(\ell^2)$ となる．また復元処理の回数が $O(\ell)$ となる．したがって表 1 からも分かるように，計算量は漸近的に LSBs Gate が優位となるが，1 節で挙げた既存のマルチパーティ計算[4)–9),14)] は何れも，乗算の秘匿関数計算が計算量，通信量ともに支配的であるため，ℓ が現実的な数値であれば，計算量の優劣はマルチパーティ計算の方式に大きく依存することが予想される．この問題は $\mathbb{Z}/m\mathbb{Z}$ 上の乗算と冪乗剰余演算の計算量の比率等が影響するため，今後実装により明らかにしていきたい．

次に，LSBs Gate および提案方式における秘匿関数計算の通信回数および通信量を比較する (表 2)．提案方式の通信回数および通信量はそれぞれ $O(\ell)$, $O(n\ell^2)$ となり LSBs Gate に等しい．ただし先述のとおり，乗算の秘匿関数計算は通信量が大きく，その点において提案方式は LSBs Gate よりも優れる．例えば文献 8) のマルチパーティ計算を用いた場合，$\ell = \lceil \log N^2 \rceil$ (N: RSA 合成数) となり，文献 8) 8.1.2 節によれば，1 回の乗算に $5.5n\ell$ ビットの通信が発生すると見積もることができる．したがって，3 節の例より乗算回数を $2\ell - 1$ とすれば，乗算に必要な通信量は $11n\ell^2 - 5.5n\ell$ ビットとなり，$n = 3$, $\ell = 2,048$ とすれば，およそ 17 MBytes となる．一方，復元に必要な通信量，すなわち提案方式の通信量は，文献 15) 3.2 節および 5.1 節より $2n\ell^2$ ビットと見積もることができ，LSBs Gate と比べ 80% 以上の削減となっていることが分かる．

表 1 LSBs Gate[12)] および提案方式における秘匿関数計算の実行回数の比較

	LSBs Gate	提案方式
乗算	$O(\ell)$	0
復元	$O(1)$	$O(\ell)$
加減算・定数倍	$O(\ell)$	$O(\ell^2)$

表 2 LSBs Gate[12)] および提案方式における秘匿関数計算の通信回数および通信量の比較

	LSBs Gate	提案方式
通信回数	$O(\ell)$	$O(\ell)$
通信量 (乗算)	$O(n\ell^2)$	0
通信量 (復元)	$O(n\ell)$	$O(n\ell^2)$

6. ま と め

マルチパーティ計算の計算量や通信量を削減できる有用なツールとして知られている，ビット分解プロトコルの処理効率向上を実現する手法を提案した．提案方式は，一般にマルチパーティ計算の計算量や通信量において支配的となる，乗算の秘匿関数計算を必要としないことが大きな特徴である．特に通信量は一般に優れ，特定のマルチパーティ計算において 80% 以上の削減効果があることを示した．また提案方式は，m を任意の整数として環 $\mathbb{Z}/m\mathbb{Z}$ 上で計算が可能である．これにより，従来仮定してきた素体上や乗法群上ではなく，m を 2 の冪乗として剰余演算を効率良く処理するといった実装上の効果も期待できる．

© 2010 Information Processing Society of Japan

Vol.2010-CSEC-50 No.45
2010/7/2

参 考 文 献

1) A. C. Yao, Protocols for secure computations, FOCS '82, pp. 160–164, IEEE Press, 1982.
2) B. Pinkas, T. Schneider, N.P. Smart, and S.C. Williams, Secure two-party computation is practical, ASIACRYPT 2009, LNCS 5912, pp. 250–267, Springer-Verlag, 2009.
3) A. C. Yao, How to generate and exchange secrets, FOCS '86, pp. 162–167, IEEE Press, 1986.
4) O. Goldreich, S. Micali, and A. Wigderson, How to play any mental game, or a completeness theorem for protocols with honest majority, STOC '87, pp. 218–229, ACM Press, 1987.
5) M. Jakobsson and A. Juels, Mix and match: secure function evaluation via ciphertexts, ASIACRYPT 2000, LNCS 1976, pp. 162–177, Springer-Verlag, 2000.
6) M. Ben-Or, S. Goldwasser, and A. Wigderson, Completeness theorems for non-cryptographic fault-tolerant distributed computation, STOC '88, pp. 1–10, ACM Press, 1988.
7) D. Chaum, C. Crepeau, and I. Damgård, Multiparty unconditionally secure protocols, STOC '88, pp. 11–19, ACM Press, 1988.
8) R. Cramer, I. Damgård, and J. B. Nielsen, Multiparty computation from threshold homomorphic encryption, EUROCRYPT 2001, LNCS 2045, pp. 280–300, Springer-Verlag, 2001.
9) B. Schoenmakers and P. Tuyls, Practical two-party computation based on the conditional gate, ASIACRYPT 2004, LNCS 3329, pp. 119–136, Springer-Verlag, 2004.
10) J. Algesheimer, J. Camenisch, and V. Shoup, Efficient computation modulo a shared secret with application to the generation of shared safe-prime products, CRYPTO 2002, LNCS 2442, pp. 417–432, Springer-Verlag, 2002.
11) I. Damgård, M. Fitzi, E. Kiltz, J. B. Nielsen, and T. Toft, Unconditionally secure constant-rounds multi-party computation for equality, comparison bits and exponentiation, TCC 2006, LNCS 3876, pp. 285–304, Springer-Verlag, 2006.
12) B. Schoenmakers and P. Tuyls, Efficient binary conversion for Paillier encrypted values, EUROCRYPT 2006, LNCS 4004, pp. 522–537, Springer-Verlag, 2006.
13) T. Nishide and K. Ohta, Multiparty computation for interval, equality, and comparison without bit-decomposition protocol, PKC 2007, LNCS 4450, pp. 343–360, Springer-Verlag, 2007.
14) 千田 浩司, 五十嵐 大, 高橋 克巳：効率的な 3 パーティ秘匿関数計算の提案とその運用モデルの考察, 情報処理学会研究報告, Vol. 2009-CSEC-48, No. 1, pp. 1–7, 2010.
15) P. Fouque, G. Poupard, and J. Stern, Sharing decryption in the context of voting or lotteries, Financial Cryptography 2000, LNCS 1962, pp. 90–104, Springer-Verlag, 2000.

© 2010 Information Processing Society of Japan