L-013

Graphics Processing Unit を用いた拡大体上の QUAD ストリーム暗号の高速化
Accelerating QUAD Stream Cipher over Extend Field using Graphics Processing Unit

田中哲士 *† Satoshi Tanaka　安田貴徳 * Takanori Yasuda　櫻井幸一 *† Kouichi Sakurai

概要

QUAD ストリーム暗号は鍵ストリームの生成に多変数公開鍵暗号の技術を利用しており，他のストリーム暗号と異なり公開鍵暗号レベルの証明可能安全性を持つ．一方で，QUAD は鍵の生成に多量の計算が必要であり，ストリーム暗号としては低速である．高速な実装手法の一つとして計算速度の高い Graphics Processing Unit (GPU) を用いて実装する方法が知られており，暗号分野においても GPU を利用した AES の効率化等が報告されている [4]．我々は，これまで QUAD の高速化に対して挑戦しており，多項式の評価に必要な計算を並列化することにより効率化を図ってきた [6]．一方で，$GF(2^{32})$ のような拡大体を用いることによる変数の削減を利用した高速化についても注目しており，効率的な拡大体上の乗算手法についても検討している [7]．本論文では GPU を用いた $GF(2^{32})$ 上の QUAD ストリーム暗号の実装を行い，その結果について述べる．

1　多変数多項式

p を素数とし，$q = p^k$ $(k \geq 1)$ とする．このとき，$GF(q)$ は標数 q の k 次拡大体とする．$GF(q)$ 上の n 個組の変数を $X = \{x_1, \ldots, x_n\}$ とすれば，$GF(q)$ 上の n 変数の二次多項式は次のように表現できる．

$$f(X(= x_1, \ldots, x_n)) = \sum_{1 \leq i \leq j \leq n} \alpha_{i,j} x_i x_j + \sum_{1 \leq i \leq n} \beta_i x_i + \gamma \quad (\alpha_{i,j}, \beta_i, \gamma \in GF(q), (1 \leq i \leq j \leq n)). \quad (1)$$

更に，$GF(q)$ 上の m 連立 n 変数二次多項式 $S(X)$ を次のように表現する．

$$S(X) = \{f_1(X), \ldots, f_m(X)\}. \quad (2)$$

2　ストリーム暗号

あるメッセージ $M = \{m_1, m_2, \ldots, m_L\}$ に対して，鍵ストリーム列 $K = \{k_1, k_2, \ldots, k_L\}$ を用いて，

$$c_i = m_i \bigoplus k_i \ \ (1 \leq i \leq L), \quad (3)$$

となる暗号文 $C = \{c_1, c_2, \ldots, c_L\}$ を生成する方式をストリーム暗号と呼ぶ．復号は同様に鍵 K を用いて，

$$m_i = c_i \bigoplus k_i \ \ (1 \leq i \leq L), \quad (4)$$

として入手することができる．メッセージ長以上の長さを持つ鍵を 1 回の暗号化にのみ使用する場合，情報理論的安全性を持つ暗号が構成されることが知られている．しかし実際には秘密裏にメッセージ長以上の暗号鍵を共有することは困難である．その為実用上は，擬似乱数生成器と乱数のシードを共有することにより，同じ鍵ストリームを生成し暗号鍵を共有する手段などが行われている．

* 九州先端科学技術研究所，Institute of Systems, Information Technologies and Nanotechnologies
† 九州大学，Kyushu University, Fukuoka, Japan

2.1　QUAD ストリーム暗号

QUAD ストリーム暗号は式 2 の結果を鍵ストリームの生成に利用するストリーム暗号である [3]．$Y = \{y_1, \ldots, y_m\} \in GF(q)^m$ を式 2 で与えられる多項式 $S(X)$ の値とする．即ち，

$$y_i = f_i(X) \ \ (1 \leq i \leq m), \quad (5)$$

である．ここで，$m = kn$ $(k \leq 2)$ とする．QUAD では $Y = \{y_1, \ldots, y_m\}$ を n 個の組 $S_{it} = \{x_1, \ldots, x_n\}$ と $(m-n) = (k-1)n$ 個の組 $S_{out} = \{x_{n+1}, \ldots, x_m\}$ の二つに分け，S_{out} を鍵ストリームとして出力する．一方で，S_{it} を次のステップにおける X として利用し，これを繰り返す．図 1 に QUAD ストリーム暗号の鍵生成のイメージを示す．

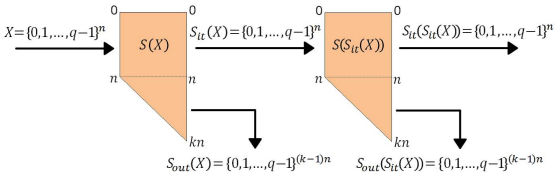

Figure 1: QUAD の鍵生成

QUAD は安全性の根拠に MQ 問題と呼ばれる数学の問題を利用している．MQ 問題は有限体上の連立二次方程式の求解問題，即ち，$GF(q)$ 上の多項式 $S(X)$ とその値 $Y = S(X)$ が与えられたとき，元の変数 X の値を求める問題である．一般に，MQ 問題は NP 完全であることが知られており [1]，QUAD は MQ 問題が解決困難である限り安全な暗号であることが証明されている．

2.2　QUAD の計算コスト

QUAD の鍵ストリーム生成では一回の鍵生成ごとに Y の計算，式 2 で与えられる多項式 $S(X)$ の評価が必要となる．$S(X)$ 内の n 変数二次多項式 $f_i(X)$ $(1 \leq i \leq m)$ は $\binom{n+1}{2}$ 個の二次の項と n 個の 1 次の項，1 個の定数項を持っている．従って，$f_i(X)$ の評価には $2\binom{n+1}{2} + n = n(n+2)$ 回の $GF(q)$ 上乗算及び $\binom{n+1}{2} + n = n(n+3)/2$ 回の $GF(q)$ 上加算が必要となり，$S(X)$ の評価には $mn(n+2)$ 回の乗算と $mn(n+3)/2$ 回の加算が必要である．

Berbain らは QUAD の計算コストを減少させる以下の手法を提案している [2]．

単項式 $x_i x_j$ の事前計算　二次の項 $x_i x_j$ $(1 \leq i \leq j \leq n)$ は各方程式で共通である．従って，この計算を事前に行うことによって全体の乗算回数を $mn(n+5)/2 + n(n+1)/2$ に減らすことができる．

ビットスライス　通常，一つの変数は一つのデータしか取り扱わないが，ビットスライスを利用し変数内に複数のデータを格納し，縦にデータを取り扱うことで変数単位で SIMD の計算が可能となり，全体の計算量を減らすことに繋がる．ビットスライス方式を適用するためには計算が並列化可能であることが重要だが，$S(X)$ 内の $f_i(X)$ はそれぞれ独立であり，並列性が高くビットスライスによる計算削減の効果は高い．

3 GPU による QUAD の実装

3.1 GPGPU 技術

画面の出力や画像処理，3DCG の演算を行うことを目的とした計算装置を Graphics Processing Unit(GPU) と呼ぶ．GPU を画像処理以外の計算に利用し高い計算能力を達成する技術を GPGPU と呼ぶ．現在は，GPU を提供する AMD や NVIDIA が提供する GPU 向けの開発キット [8, 9] や OpenCL 等のライブラリを用いることで，GPU 環境におけるプログラム開発が可能となってきている．

3.2 QUAD ストリーム暗号の並列化

現在の GPU は 1,000 個以上のコアを有しており，CPU よりも大きな計算能力を有しているといえる．一方で，GPU の各コアの計算能力は CPU のコアよりも低い．その為，GPU 上で実装を行う場合，その並列性を利用し同時に多数の計算を行うことが望ましい．QUAD の場合，以下の 2 点の並列化が必要である．

1. 連立多項式 $Q(X) = \{f_1(X), \ldots, f_m(X)\}$ の並列化
2. 二次多項式 $y_i = f_i(X)$ の効率化

我々は以下の戦略で並列化を試みた．第 1 に連立多項式の並列化については berbain らのビットスライス [2] を採用した．$S(X)$ 内の多項式 $f_i(X)$ はそれぞれ独立であり並列化の効果が高いためである．第 2 に各多項式の計算は以下の 2 ステップで計算を行った．第 2 ステップの並列リダクションについては [6] のものを用いた．

1. 二次多項式 $y_i = f_i(X)$ 内の全ての項 $\alpha_{j,k}^{(i)} x_j x_k$, $\beta_j^{(i)} x_j$ を事前に計算する．
2. 総和 $\sum_{1 \le j \le k \le n} \alpha_{i,j}^{(i)} x_j x_k + \sum_{1 \le j \le n} \beta_j^{(i)} x_j + \gamma^{(i)}$ を並列リダクションを用いて求める．

4 実装実験

実装実験として，$GF(2^{32})$ 上の 32, 48, 64 変数（$k = 2$) の QUAD を CPU 及び GPU 上で実装し，それぞれ，10MB のデータの暗号化時間を測定した．実装結果について，表 1 に示す．また，実装環境として，GPU に NVIDIA GeForce GTX TITANCPU に Intel Core i7 875K を利用した．メモリ環境は 8GB である．

Table 1: GPU を用いた $GF(2^{32})$ 上の QUAD 実装結果

変数	32	48	64
多項式	64	96	128
CPU			
暗号化時間 (sec)	205.105	298.842	392.277
スループット (kbps)	49.926	34.266	26.104
GPU			
暗号化時間 (sec)	11.718	10.068	18.000
スループット (kbps)	873.869	1017.084	568.889
速度比	×17.503	×29.683	×21.793

5 まとめ

本論文では，QUAD ストリーム暗号の並列化手法について述べ，GPU 上で実装した結果を述べた．我々の実装手法では $GF(2^{48})$ 上の 48 変数の QUAD ストリーム暗号を GPU 上で実装することによって，約 30 倍の高速化を実現することに成功した．

今後の課題として，Petzoldt の手法 [5] の適用を検討する．Petzoldt は Linear Recurring Sequences を利用する事により，乗算の計算量を $m \cdot n + m$ 回に落とすことに成功しており，高速な手法として期待できる．本発表では，Petzoldt の手法の適用結果について発表する．

また，実験結果では 48 変数から 64 変数の時点で速度比 29.7 倍から 21.8 倍に下がっている．これは，総和等の一部計算で多数のスレッドが必要な計算であり，NVIDIA GeForce GTX TITAN の 1 ブロックで取り扱えるスレッド数を超えていたと考えている．今後，QUAD の動作についてより詳しく分析し推測が正しいかを検証する．

謝辞

本研究は一部において，総務省戦略的情報通信研究開発推進事業（SCOPE）平成 25 年度 ICT イノベーション創出型研究開発フェーズ I『多変数多項式システムを用いた安全な暗号技術の研究』（課題番号 0159-0172）の助成を受けている．

References

[1] Bard, G.V.: NP-Completeness of MP, In Algebraic Cryptanalysis, pp. 199-202, Springer, 2009

[2] Berbain C., Billet, O., Gilbert, H.: Efficient Implementations of Multivariate Quadratic Systems. In the Proceedings of Selected Areas in Cryptography, LNCS, vol. 4356, pp. 174-187, Springer, 2007.

[3] Berbain, B., Gilbert, H., Pataring, J.: QUAD: a Pratical Stream Cipher with Provable Security. In EUROCRYPTO'06, LNCS, vol.4004, pp.109-128, Springer, 2006.

[4] Manavski, S.A.: Cuda compatible gpu as an efficient hardware accelerator for aes cryptography. In International Conference on Signal Processing and Communications 2007 (ICSPC 2007), pp. 65-68. IEEE, 2007.

[5] Petzoldt, A.: Speeding up QUAD. In cryptography ePrint archive, report, 2013.

[6] Tanaka, S., Nishide, T., Sakurai, K.: Efficient Implementation for QUAD Stream Cipher with GPUs., In the Journal of Computer Science an d Information Systems (ComSIS), vol. 10, num. 2, special issue, pp.897-911, 2013.

[7] Tanaka, S., Yasuda, T. Sakurai, K.: Implementation of Efficient Operations over $GF(2^{32})$ using Graphics Processing Units. In Information and Communication Technology, LNCS, vol. 8407, pp.602-611, Springer, 2014.

[8] Graphics Development
`http://developer.amd.com/tools-and-sdks/graphics-development/`

[9] CUDA Zone
`https://developer.nvidia.com/cuda-zone`