認証付き暗号 Minalpher の GPGPU 実装

小杉真紀子[1]　佐藤証[1]

概要：認証付き暗号の標準化を進める CAESAR コンテストに日本から応募されたアルゴリズム Minalpher の GPGPU 実装評価を行った．データの機密性だけでなく完全性も保持する認証付き暗号 Minalpher は，暗号化・認証子生成の両方で同一のアルゴリズムを用いるが，このアルゴリズムは入力に依存関係がないため，並列に実行が可能である．この並列性を利用し，近年ハイパフォーマンスコンピューティングの分野で広く利用されている GPGPU を用いて Minalpher を実装した．現在認証付き暗号の事実上の標準アルゴリズムとして利用されている AES-GCM と処理速度を比較し，速度性能における Minalpher の優位性を示した．

Software Implementation of Authenticated Encryption Algorithm Minalpher on GPGPU

MAKIKO KOSUGI[1]　AKASHI SATOH[1]

1. はじめに

暗号化によってデータの守秘性は確保できるが，アルゴリズムによっては暗号文の内容がわからなくても，一部の書き換えや全体を置き換えることが可能な場合がある．そこで暗号化と認証を同時に行う認証付き暗号の標準化が活発化し，米国標準技術研究所 (NIST : National Institute of Standards and Technology)が支援する CAESAR (Competition for Authenticated Encryption: Security, Applicability, and Robustness)プロジェクトが 2013 年 1 月に開始された[1]．認証付き暗号の標準としては，AES-GCM (Galois Counter Mode)[2]が広く利用されているが，これよりも優れたアルゴリズムの標準化を目的に，世界中から提案された 57 のアルゴリズムの安全性評価と，ソフトウェアおよびハードウェアの実装評価が続いており，4 回の国際会議を経て，2017 年 12 月に標準アルゴリズムが決定される予定である．

Minalpher[3]は，三菱電機，NTT，福井大学が開発し，CAESAR に日本から応募されたアルゴリズムである．筆者らは既に Minalpher を FPGA 上に実装し，現在認証付き暗号の事実上の標準である AES-GCM とその性能を比較することで Minalpher の優位性を示した[4]．そこで，本稿では GPGPU (General-Purpose computing on Graphic Processing Units)への実装を行い，現在認証付き暗号の事実上の標準アルゴリズムとして利用されている AES-GCM と速度性能を比較して，その優位性を示す．

2. GPGPU

GPGPU は，本来画像処理のプロセッサである GPU を，画像処理以外の目的に応用する技術のことである．GPU は画像処理の特性上，計算能力が低いコアを大量に有しており，単純な処理を同時に大量に行う計算に向いている．そのため，近年ではハイパフォーマンスコンピューティングの分野で多く利用されている．

GPGPU が始まった当初は，OpenGL 等グラフィックス用の言語を使ったプログラミングが主流であった．しかし，対応する命令が少ないことや，開発環境の整備等，汎用計算を行う上で問題が多く存在した．2007 年に GPU の大手ベンダーである NVIDIA 社が，NVIDIA 社製の GPU を用いた並列コンピューティングのための統合開発環境 CUDA (Compute Unified Device Architecture)[5][6]の提供を開始した．

現在 GPU 内部には，CUDA コアと呼ばれる演算を行うコアが，ストリーミングマルチプロセッサ(Streaming Multi-processor：SM)として CUDA コアを複数個まとめられている．この CUDA コアや SM の数は GPU の世代によって異なるが，NVIDIA 社の最新 GPU である GeForce GTX 980 Ti では，CUDA コアが 2,816 基，Maxwell SM (通称 SMM)が 22 基用意されている．

CUDA では，並列に実行する関数(カーネル関数)を定義し，それを最小の計算単位であるスレッドで実行する．スレッドはブロック(本稿ではスレッドブロックと表記する)に複数個まとめられ，ブロックは複数個にまとめられてグ

1 電気通信大学大学院情報理工学研究科

リッドとして管理される．同じスレッドブロック内のスレッドは同じ SM 内で動作し，各スレッドは 1 つの CUDA コアで処理される．スケジューラが CUDA コアに処理するスレッドを割り当てることで，CUDA コアのコア数を超えるスレッド数の計算でも，並列に処理しているように実行される．

3. 先行研究

暗号分野における GPGPU の利用は，2005 年頃には行われていた．Cook らは，GPGPU で AES を実装した研究を 2005 年に発表した[7]．当時は CUDA が提供される前であり，OpenGL を使って実装された．実装は NVIDIA 社の GeForce3 を使って行われたが，OpenGL には排他的論理演算等の暗号アルゴリズムに必要な演算をサポートする API が少なかった等の理由で, GPGPU を使った際のスループットが 1.53Mbps と，CPU を使った場合の最大で 2.5%のスループットしか出なかった．このため，GPGPU は暗号の実装には適切ではないとされた．しかし，CUDA が公表された 2007 年，Manavski が CUDA を利用して AES を GPGPU で実装したと発表[8]．NVIDIA 社の GeForce8800 GTX を使用した 128 ビット AES の実装では, メモリ転送時間を含めた速度性能が，汎用プロセッサから最大で 5.92 倍向上し，暗号分野における GPGPU の有効性を示した．

4. Minalpher アルゴリズム

Minalpher は共通鍵暗号方式のブロック暗号である．表 1 のデータを扱い，暗号化・復号と同時に認証子の生成・検証を行う AEAD (Authenticated Encryption with Associated Data)モードと，認証子の生成・検証だけを行う MAC (Massage Authentication Code)モードを実現する．これらの暗号化処理時のデータの流れを図 1 に示す．図中の P は Minalpher の 256 ビットの非線形置換関数であり, これについては次節で詳解する．また，図 1 中の γ，φ，ψ は鍵，ナンスおよび定数 *flag* より生成される変数であり，それぞれの生成手順は 4.2 節で示す．

Minalpher は，入力するデータを切り替えることにより，暗号化と復号，そして認証子の生成と検証で同じ関数を使用できる．また，AEAD モードと MAC モードも同様である．暗号化・復号と認証子生成・検証には同じ関数 P が繰り返し用いられるが，1 つの P 関数の回路ブロックを使い回したり，図 1 で入力データに依存関係が存在しないことを利用して複数の P 関数ブロックを実装して並列処理するなど，小型化や高速化が可能となる．

表 1　Minalpher の入出力データ

データ		入力	出力	ビット幅
鍵 K	暗号化	あり	なし	128
	復号			
ナンス N	暗号化	あり	なし	104
	復号			
Associated Data N	暗号化	あり	なし	256 ビットごとに分割し処理
	復号			
平文 M	暗号化	あり	なし	256 ビットごとに分割し処理
	復号	なし	あり	
暗号文 C	暗号化	なし	あり	256 ビットごとに分割し処理
	復号	あり	なし	
認証子 *tag*	暗号化	なし	あり	128
	復号	あり	あり	

4.1 置換関数 P

置換関数 P は，256 ビットの 4 つの基本関数 S，T，M，E で構成されており，256 ビット入力 X_{in} に対して，S，T，M，E の組を 18 ラウンド繰り返して X_{out} を出力するが，最終ラウンドは S と T だけを実行し，M と E はスキップする．復号で用いる逆関数 P^{-1} は P と同じになるよう工夫されている．なお，S は 8 ビット入出力の S-box を用いた置換，T は 32 ビットごとの ShuffleRows，M は 32 ビットごとの MixColumns，そして E はラウンド数 i から生成された 128 ビットの定数 RC_i と排他的論理演算を行う関数である．

4.2 変数生成アルゴリズム

256 ビットの変数 γ，φ，ψ は，ブロックが入力されるごとに更新され，置換関数 P の前後で，P の入力・出力ブロックと排他的論理演算を行う変数である．この生成アルゴリズムは，256 ビットの L または L'の生成と，TweakUpdate と呼ばれる処理の繰り返しで構成される．入力された 128 ビットの鍵 K，24 ビットの定数 *flag*，入力された 104 ビットのナンス N を連結し，256 ビットの変数とし，次式により関数 P を用いて L を生成する．

$$L = (K \,||\, flag \,||\, N) \oplus P(K \,||\, flag \,||\, N) \qquad (1)$$

TweakUpdate では，32 バイト(256 ビット)入力$y_{31} \ldots y_1 y_0$, 32 バイト出力を$x_{31}x_{30} \ldots x_1 x_0$とするとき, 式(2)(3)の処理を行う．

$$\begin{cases} x_{31}x_{30} \ldots x_5 x_4 = y_{30}y_{29} \ldots y_4 y_3 \\ x_3 = y_2 \oplus y_{31} \\ x_2 = y_1 \oplus y_{31} \\ x_1 = y_0 \end{cases} \qquad (2)$$

1 バイトの変数y_{31}とx_{31}をそれぞれ$y_{31} = b_7 b_6 \ldots b_1 b_0$, $x_{31} = a_7 a_6 \ldots a_1 a_0$と表すとき，

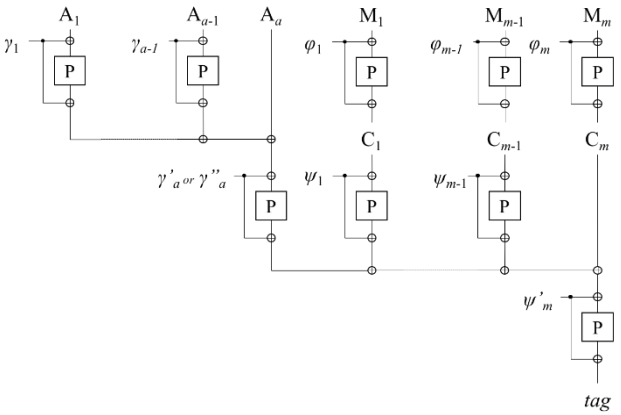

図 1　Minalpher のデータの流れ

$$\begin{cases} a_7 = b_6 \oplus b_7 \\ a_6 = b_5 \\ a_5 = b_4 \oplus b_7 \\ a_4 a_3 a_2 = b_3 b_2 b_1 \\ a_1 = b_0 \oplus b_7 \\ a_0 = b_7 \end{cases} \quad (3)$$

そして 256 ビットの各変数 γ，φ，ψ は，TweakUpdate により次式で生成する．

$$\begin{cases} \gamma_i = \begin{cases} \mathrm{TweakUpdate}(L') & \cdots (i = 1) \\ \mathrm{TweakUpdate}(\gamma_{i-1}) & \cdots (2 \le i \le a-1) \end{cases} \\ \gamma'_a = \gamma_{a-1} \cdot \mathrm{TweakUpdate}(\gamma_{a-1}) \\ \gamma''_a = \gamma'_a \cdot \mathrm{TweakUpdate}(\gamma'_a) \end{cases} \quad (4)$$

$$\varphi_j = \begin{cases} \mathrm{TweakUpdate}(L) & \cdots (j = 1) \\ \mathrm{TweakUpdate}\big(\mathrm{TweakUpdate}(\varphi_{j-1})\big) & \cdots (2 \le j \le m) \end{cases} \quad (5)$$

$$\begin{cases} \psi_j = \mathrm{TweakUpdate}\Big(\mathrm{TweakUpdate}\big(\psi_{j-1}\big)\Big) \\ \qquad\qquad \cdots (1 \le j \le m-1) \\ \psi'_m = \psi_{m-1} \cdot \mathrm{TweakUpdate}(\psi_m) \end{cases} \quad (6)$$

5. 実装

Minalpher および AES-GCM のアルゴリズムを実装した．

Minalpher に関して，本研究では 256 ビットのデータを 1 つのデータブロックとして，P 関数とその前後に行われる変数との排他的論理和を 1 つのカーネル関数として定義し，暗号化部および認証子生成部で行われる処理のうち，認証子生成部の A_{a-1} と C_{m-1} 以外の処理を並列化した．Associated Data に関する認証子生成を行ったのち，平文の暗号化と生成された暗号文に関する認証子生成を行った．

AES-GCM は参考文献[9]のアルゴリズムを使用した．AES-GCM は AES-CTR (Counter)[9]で暗号化を行う．この AES-CTR は並列処理が可能であることが知られている．本研究では 128 ビットのデータを 1 つのデータブロックとして，AES-CTR の 1 ブロックに対する暗号化処理を 1 つのカーネル関数として定義した．

GPGPU の開発環境は表 2 のとおりである．

表 2　開発環境

OS	Windows8.1		
Mother Board	MSI H97		
CPU	Intel Core i5		
Memory	DDR3 PC3-12800 4GB ×2		
	DDR3 PC3-12800 8GB ×1		
GPU	仕様詳細	CUDA コア	2816
		ベースクロック	1000MHz
		消費電力	250W
		バスタイプ	PCI-E 3.0
		メモリスペック	
		メモリクロック	7.0Gbps
		メモリ量	6144MB
		メモリインタフェース	384-bit GDDR5
		最大バンド幅	336.5GB/s
POWER	750W		

並列数を 13,384 として，平文 M のデータブロック数を変化させ，処理時間を計測し，Minalpher と AES-GCM の各スループットを算出し，表 3 にまとめた．なお，この算出には以下の式を用いた．

$$\frac{\text{ブロック数} \times 1\text{ブロックあたりのビット数}}{\text{処理時間}} \quad \ldots (7)$$

表 3　各アルゴリズムの計測結果

	処理時間 / ms		スループット / Mbps	
ブロック数	AES-GCM	Minalpher	AES-GCM	Minalpher
500	3.540	6.625	18.079	19.320
1000	5.876	7.068	21.783	36.222
2000	10.344	7.553	24.750	67.792
3000	14.981	8.217	25.633	93.461
4000	19.533	8.972	26.212	114.130
5000	24.009	9.589	26.657	133.481
6000	28.512	10.248	26.936	149.890
7000	33.148	11.015	27.030	162.684
8000	37.623	11.654	27.218	175.740
9000	42.219	12.302	27.286	187.281
10000	46.624	13.116	27.454	195.181
12500	58.750	14.880	27.234	215.061
15000	72.338	22.130	26.542	173.522
17500	84.447	23.863	26.526	187.738
20000	96.203	25.499	26.610	200.789
22500	107.287	27.037	26.844	213.042
25000	118.425	28.812	27.021	222.130
27500	130.657	30.628	26.941	229.857
30000	142.509	37.955	26.946	202.345
32500	154.404	39.478	26.942	210.748
35000	165.254	41.193	27.110	217.511
37500	177.059	42.746	27.110	224.583
40000	188.659	44.575	27.139	229.727

この結果より，各アルゴリズムのスループットを図 2 に示した．Minalpher はブロック数が 8,000 を過ぎた辺りから，AES-GCM はブロック数が 5,000 を過ぎた辺りからスループットが横ばいになった．Minalpher のスループットは AES-GCM よりも大きく，ブロック数 27,500 の時に最大の 8.5 倍のスループットを得ることが出来た．

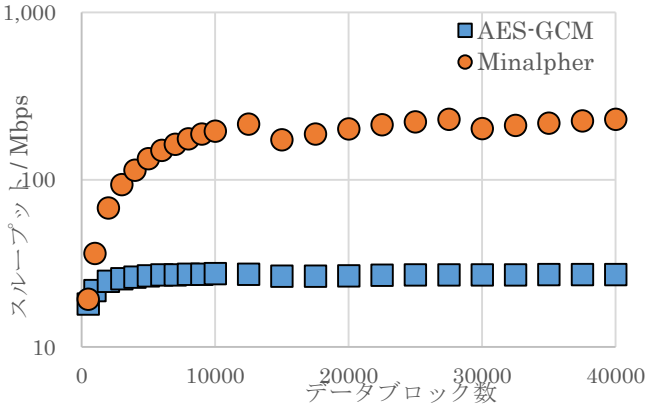

図 2　Minalpher と AES-GCM のスループット

6. 考察

Minalpher のスループットが AES-GCM を上回った要因として，Minalpher が暗号化部と認証子生成部で同じ P 関数を使用していることが挙げられる．AES-GCM は暗号化部に AES の暗号利用モードの一つである CTR モードを，認証子生成部に GHASH 関数をそれぞれ使用する．特に GHASH 関数の処理は 128 ビットのビットシフトを行う乗算処理のため，処理に時間がかかったと考えられる．一方で Minalpher が使用する P 関数は，上述の通り単純な 4 つの基本関数から成る関数のため，AES-GCM と差が出たと考えられる．ここで，AES-GCM の各処理部にかかる時間の比率を図 3(a)に示した．AES-GCM では，全処理時間の約 98%が認証子生成部のビットシフトであり，GPGPU を使って AES-GCM を実装する際における負荷が最も大きいことは数値上でも明らかである．また，AES-GCM と Minalpher の平文 M40,000 ブロックを暗号化処理する際の各処理部の処理時間及びスループットを表 4 に示した．暗号化処理に関して，AES-GCM は Minalpher よりも少ない時間で処理している．AES-GCM と Minalpher は 1 ブロックのビット数がそれぞれ 128 ビットと 256 ビットであるため，スループットは Minalpher の方が大きいが，暗号化処理部の処理速度は Minalpher との違いがより小さいと考えられる．以上より，Minalpher の処理速度における優位性は認証子生成にあると考えられる．ただし，AES-GCM の認証子生成は並列処理可能であることがわかっている[10]．現在は CPU による逐次実行で行っているが，今後はこのアルゴリズムを GPGPU に適用し，評価を行う必要があるだろう．

一方で，Minalpher の各処理部の処理時間の比率を図 3(b)に示した．Minalpher では，関数 P の処理前と処理後に行われる排他的論理和で使用する 32 バイトの変数 γ，φ，ψ の生成が処理時間全体の約 39%を占めている．この変数生成アルゴリズムは 1 バイトごとの循環型ビットシフトだが，式(2)および(3)があるため並列処理ができない．このため，変数生成アルゴリズムは CPU 側で逐次処理して生成した．その生成時間が GPU 内で並列に実行される関数 P の処理時間よりも大きいため，このような結果が出たと考えられる．

よって，AES-GCM に対する Minalpher の優位性は認証子生成における処理速度にあると言える一方，Minalpher のボトルネックは関数 P の前後に行う排他的論理和で使用する変数生成にあると考えられる．

7. むすび

認証付き暗号 Minalpher を GPGPU 上に実装した．Minalpher は入力データを独立に処理できる並列処理性があるため，その並列処理性を利用して GPGPU で実装し，現在認証付き暗号の事実上の世界標準ある AES-GCM と速度性能を比較した．その結果，最大で約 8.5 倍のスループットを得た．Minalpher は AES-GCM に対して特に認証子生成の点において有効性を示せた一方で，Mianlpher の処理上のボトルネックが認証子生成を行う際に利用する変数の生成にあることも明らかになった．

参考文献

[1] David A. McGrew, et al., “The Galois/CounterMode of Operation (GCM)”, http://siswg.net/docs/gcm_spec.pdf
[2] The international cryptologic research community, “CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness”, April 2013. http://competitions.cr.yp.to/caesar.html.
[3] Y. Sasaki, et al., “Minalpher v1.1” http://info.isl.ntt.co.jp/crypt/minalpher/files/minalpher11.pdf.
[4] Makiko Kosugi, et al, “FPGA Implementation of Authenticated Encryption Algorithm Minalpher”, GCCE 2015, 2015.
[5] NVIDIA, URL: http://www.nvidia.co.jp/object/cuda-parallel-computing-platform-jp.html.
[6] Debra L. Cook, et al., “CryptoGraphics: Secret Key Cryptography Using Graphics Cards”, Topics in Cryptology – CT-RSA 2005 Volume 3376 of the series Lecture Notes in Computer Science, pp 334-350, 2005.
[7] Svetlin A. Manavski, “CUDA Compatible GPU as an Efficient Hardware Accelerator for AES Cryptography”, Signal Processing and Communications, 2007. ICSPC 2007. IEEE International Conference on, pp 65-68, 2007.
[8] Morris Dworkin, “Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC”, NIST Special Publication 800-38D, November 2007.
[9] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Methods and Techniques", NIST Special Publication 800-38A, 2001.
[10] Akashi Satoh, “High-Speed Parallel Hardware Architecture for Galois Counter Mode”, 2007 IEEE International Symposium on Circuits and Systems, pp 1863-1866, 2007.

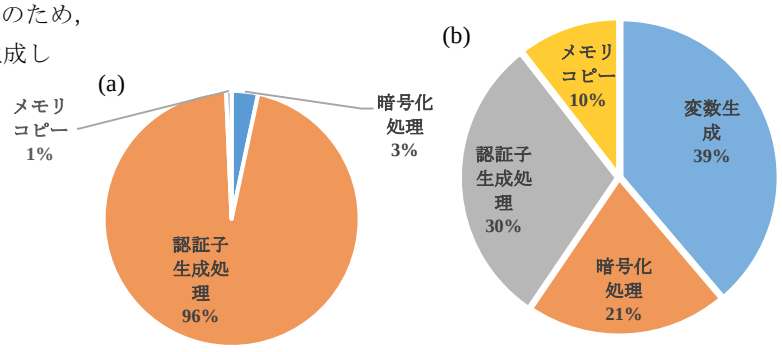

図 3　各処理部の処理時間比率 (a: AES-GCM) (b: Minalpher)

表 4　各アルゴリズムの各処理部の処理時間およびスループット

	処理時間 / ms					スループット / Gbps				
	暗号化処理	認証子生成処理	メモリコピー	メモリセット	変数生成処理	暗号化処理	認証子生成処理	メモリコピー	メモリセット	変数生成処理
Minalpher	8.079	11.705	4.104	0.050	15.143	1.267	0.875	2.495	204.224	0.676
AES-GCM	6.121	175.791	1.258	0.032		0.837	0.029	4.070	161.938	