C-009

SoC 内 GPU を用いた暗号処理を効率的に実現する抽象化層の提案
A Proposal of Cryptographic Processing System using an Abstraction Layer with GPU on SoC.

竹田 大将[1)]　杉野 栄二[1)]
Hiromasa Takeda　Eiji Sugino

1　はじめに

暗号は現在の情報社会では必要不可欠なものである．しかしながら，容量の大きなファイルやディスク全体の暗号化を行う場合，その計算負荷の高さから CPU リソースが占有され全体のパフォーマンスが低下する問題がある．主に高速化を目的としてすでに行われている GPU を用いた暗号処理の実現 [1][2][3] は，CPU リソースの占有を防ぐという観点から見ても効果的な手法であるが，その多くが NVIDIA 製のグラフィックボード上での実装を前提としており，現在普及しているラップトップ PC やシングルボードコンピュータへの適用は難しい．近年 IoT システムでは，デバイスからのデータ収集もサイバー攻撃の対象となり，シングルボードコンピュータで暗号化処理を行う必要も出てきているが，暗号化処理時間を短くするためには CPU 性能を上げねばならず，製品コストの増大をまねく．

そこで本研究では，以上の問題を解決するために System on a Chip(SoC) 内 GPU を用いたデータ暗号化を効率的に実現する抽象化層の設計・実装に取り組んでいる．現在普及しているコンピュータに搭載されている SoC の多くには GPU が内蔵されている．よって SoC 内 GPU を活用した暗号化を実現することによりグラフィックボードが搭載されないコンピュータであっても CPU リソースの占有を防ぐことが期待できる．また，暗号処理のようなシステムレベルソフトウェアを GPGPU で実現する場合 Mediator が必要とされている [4][5] が，現在 SoC 内 GPU の Mediator に関する研究は少ない．以上のことから，本システムは SoC 内 GPU を用いた暗号実装に加え，SoC 内 GPU を効率的に管理する抽象化層の実装を目指している．本稿では，システム全体の提案と SoC 内 GPU を用いた暗号実装とその評価について報告する．

2　提案システム Genc の概要

提案システム Genc(GPU encryption system) の概要図を図 1 に示す．

GPU を GPGPU 用途で扱う場合，アプリケーションごとにデバイスのセットアップが必要である．セットアップには非常に時間がかかるため，タスクを実行するたびにセットアップを行うことは非効率である．よって，初回起動時に 1 度だけセットアップを行い，その後はプロセス間通信によりタスクを待ち受ける GPU Client を常時待機させる．GPU Client はタスクを受け取るたび GPU Kernel を実行する．

次に Mediator を起動し，暗号タスクの受付を開始する．Mediator は大量に発行される暗号タスクの調停を行い，GPU に逐次的にタスクを供給する．アルゴリズムの工夫によって効率的なタスク処理を実現可能とする重要な機能となる．例えば，GPU が大容量のファイルの暗号化を得意とする特徴を踏まえ，タスクが溜まった場合関係性のあるファイルをひとまとめにパックして GPU に供給する，または一定の基準より容量が大きければ GPU に，小さければ CPU に供給先を適宜切り替えるなどの工夫が可能である．

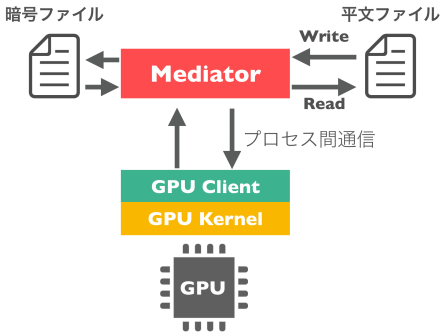

図 1　System overview

現時点では GPU Client がユーザ空間でしか動作しないため，Mediator もユーザ空間に存在する設計となっている．また，Mediator へのデータ入力は動的ライブラリ関数の上書きによって read/write システムコールを Hook している．read/write の Hook 部分は ptrace や gVisor を活用することにより，アプリごとにライブラリ関数を上書きせずに Hook できるよう検討中である．

3　SoC 内 GPU を用いた暗号実装

本節では，提案したシステムの一部である GPU Kernel 部の実装について述べる．

3.1　Intel HD Graphics

実装環境には，一般的に普及している SoC 内 GPU として Intel HD Graphis を選定した．Intel HD Graphics とは Intel プロセッサ内蔵 GPU のシリーズである．2018 年に Intel 社が公式に Intel HD Graphics 用 GPGPU 言語の C for Metal(CM) 言語を公開したことにより，GPGPU 技術の活用が可能となった．しかし，公開から日が浅いことから情報が少なく，GPGPU 用途において開発・性能効率面で有用であるか現在研究が進められている [6]．

3.2　実装暗号アルゴリズム

暗号アルゴリズムは広く普及している AES[7]，暗号利用モードは Counter(CTR) モード [8] を選定し実装した．暗号利用モードの一つである CTR モードは，1 ずつ増加していくカウンタを暗号化して鍵ストリームを生成し，生成した鍵ストリームと平文ブロックとの XOR を取った結果が暗号ブロックとなる．よって，ブロックを任意の順序で暗号化・復号できる性質があり，処理を並列化することが可能である．

1)　岩手県立大学大学院 ソフトウェア情報学研究科

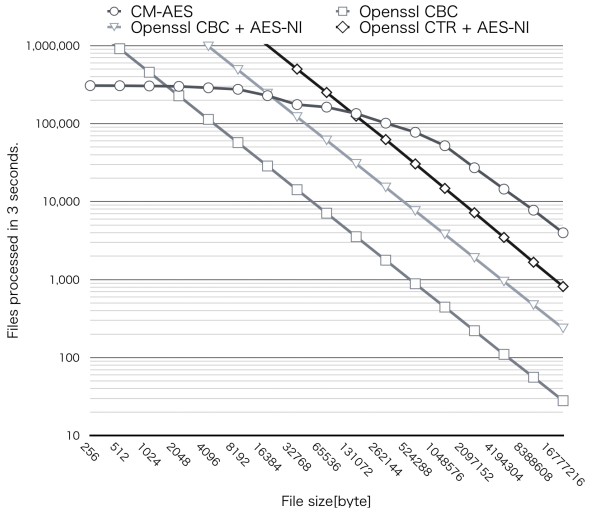

図 2　AES Performance

4　実験

4.1　実験 1

暗号アルゴリズムは AES，鍵長は 128bit，ラウンド数 10 で，256B〜16MB の複数の入力ファイルサイズを想定し実験を行い，3sec 内に暗号化できた総ファイル数で評価する．ここに復号は含めない．また，OpenSSL に内蔵されている最適化された AES 実装をそれぞれ，以下の 3 つのパターンを同 SoC 内 CPU（i3-8109U）で実行・測定し比較する．

- 現在最も普及しているが暗号化において並列化が行えない特徴を持つ CBC モード実行
- Intel チップの AES-NI 拡張命令セットを使用した CBC モード実行
- Intel チップの AES-NI 拡張命令セットを使用した CTR モード実行

結果のグラフを図 2 に示す．本研究で実装したカーネルが「CM-AES」である．

この結果から，広く普及している CPU 実装の CBC モードとの比較においては，AES-NI 拡張命令セットを使用しない場合 2kByte あたりから，AES-NI 拡張命令セットを使用した場合でも 32kByte あたりから CM-AES の性能が上回っていることがわかる．また，CPU 実装の CTR モードと比較しても 128kByte あたりから CM-AES の性能が上回っている．データサイズが小さすぎる場合に CPU に比べ GPU 速度が劣る原因は，GPU の計算処理にかかる時間に比べデータの転送に要する時間が大きく，転送時間を隠蔽することができないためであると推測される．

4.2　実験 2

本研究で実装した Kernel 部と OpenSSL の AES それぞれを CPU に負荷をかけた状態で実行し，スループットの変化を比較する．スループットが大きく変わらなければ CPU を占有していない証明となる．GPU の使用により CPU の占有が抑えられることを示すことがこの実験の目的である．

表 1　Measurement of throughput change by stress command

	CPU 負荷なし (Files)	CPU 負荷あり (Files)	性能低下率 (%)
OpenSSL CTR + AES-NI	815	464	約 43 %
CM-AES	3,625	3,556	約 2 %

4.2.1　実験方法

ファイルサイズ 16,777,216Byte の時，3 秒以内に暗号化を完了したファイル数を以下の 2 パターンで計測する．

- CPU に負荷をかけない状態で実行
- stress コマンドにより CPU の全コアに負荷がかかった状態で実行

比較対象は同 SoC 内 CPU（i3-8109U），動作周波数はプロセッサー・ベース動作周波数である 3GHz に固定する．

4.2.2　実験結果

実験結果を表 1 に示す．CM-AES は OpenSSL ほどスループットが低下していないことがわかる．よって，CM-AES は CPU の占有が抑えられることが明らかになった．

5　おわりに

本稿では，SoC 内 GPU を用いた暗号処理を効率的に実現する抽象化層のシステム全体の提案と，その一部機能である AES 暗号の SoC 内 GPU 実装を示した．実験により，暗号化するデータサイズが 2kByte 以上でないと CPU 実装に比べ十分な効果が得られないことがわかった．従って，データサイズにより CPU・GPU を使い分けることで処理効率を向上することが期待できる．他にアプリケーションの用途によって CPU リソースを開放するスケジューリングも検討していく予定である．

参考文献

[1] Sakai, Ryosuke, Koji Nakano, and Yasuaki Ito : “Accelerating RSA encryption using GPUs.”Bulletin of Networking, Computing, Systems, and Software 4.1 pp.69-73 (2015).
[2] SAXENA, Aryan, et al: “Accelerating Image Encryption with AES Using GPU: A Quantitative Analysis.”International Conference on Intelligent Systems Design and Applications. Springer, Cham, pp.372-380 (2018).
[3] WANG, Wei, et al:“Accelerating fully homomorphic encryption using GPU.”2012 IEEE conference on high performance extreme computing. IEEE, pp.1-5 (2012).
[4] Christopher J. Rossbach, Jon Currey, Mark Silberstein, Baishakhi Ray, Emmett Witchel: “PTask: Operating System Abstractions To Manage GPUs as Compute Devices,”SOSP, (2011).
[5] Kato, Shinpei, et al: “Gdev: First-Class GPU Resource Management in the Operating System.”USENIX, (2012).
[6] 近藤 鯛貴, 竹田 大将, 佐藤 裕幸: “C for Metal 言語を用いた Intel HD Graphics の GPGPU 用途における性能評価”2019-HPC-172, pp.1-6 (2019).
[7]“Announcing the ADVANCED ENCRYPTION STANDARD (AES)”NIST, (2001).
[8] Morris Dworkin: “Recommendation for Block Cipher Modes of Operation”NIST, (2001).