Implementasi AES-ECB 128-bit untuk Komputasi Paralel pada GPU menggunakan Framework NVIDIA CUDA

Adityo Jiwandono, 13507015[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*jiwandono@arc.itb.ac.id*

***Abstract*— Teknologi Graphics Processing Unit (GPU) telah berkembang pesat. Penggunaan GPU sekarang tidak hanya di bidang grafis saja, namun sudah merambah ke bidang yang umum. GPU memiliki kemampuan komputasi paralel yang tinggi persis seperti kebutuhan pemrosesan grafis. Di sisi lain, kebutuhan akan pertukaran data digital yang aman dan cepat semakin meningkat. Implementasi algoritma kriptografi AES-ECB pada platform komputasi paralel diharapkan dapat meningkatkan performa komputasi pada AES itu sendiri. Makalah ini akan membahas bagaimana algoritma AES-ECB dapat diimplementasikan pada GPU dengan mengeksploitasi fitur-fitur dan karakteristik komputasi paralel yang dimiliki oleh GPU. Pada makalah ini juga akan dibandingkan perbedaan kecepatan pemrosesan untuk algoritma kriptografi AES-ECB 128-bit yang diimplementasikan pada CPU (OpenSSL) dan GPU.**

***Kata kunci:* AES, block cipher, komputasi paralel, kriptografi simetri.**

I. PENDAHULUAN

Kebutuhan akan solusi kriptografi yang efisien makin meningkat dalam satu dekade ini. Peningkatan kebutuhan ini disebabkan oleh penggunaan Internet untuk keperluan yang kritis seperti bisnis, pemerintahan, dan kesehatan. Solusi untuk mengakselerasi performa kriptografi yang berbasiskan perangkat lunak maupun perangkat keras sudah banyak dipelajari dan diusulkan baik di dunia akademis maupun industri.

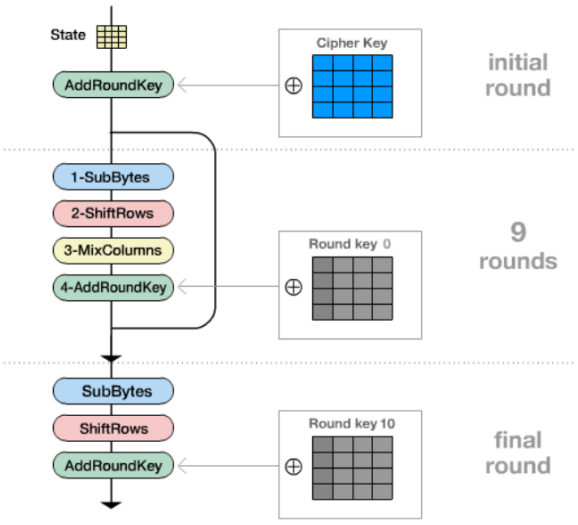
Salah satu solusi untuk mengakselerasi performa kriptografi menggunakan perangkat keras adalah dengan memanfaatkan kemampuan komputasi GPU. Performa GPU meningkat pesat dan bahkan dapat mengalahkan performa CPU dalam hal komputasi *floating-point*. Hal ini bisa terjadi karena GPU memang didesain untuk komputasi yang intensif sesuai dengan kebutuhan aslinya yaitu grafis.

Pada November 2006, NVIDIA memperkenalkan teknologi *Compute Unified Device Architecture* (CUDA). CUDA adalah arsitektur komputasi paralel untuk tujuan yang umum yang meningkatkan kemampuan mesin komputasi paralel dalam GPU NVIDIA untuk dapat menyelesaikan persoalan-persoalan komputasi yang kompleks secara lebih efisien daripada CPU. CUDA memiliki model pemrograman dan sekumpulan arsitektur instruksi yang baru.

Algoritma *Advanced Encryption Standard* (AES) dipilih dalam makalah ini karena AES merupakan salah satu standar algoritma kriptografi yang digunakan secara luas. Selain itu, algoritma AES sangat ringkas dan teknik-teknik implementasinya didokumentasikan dengan baik. Panjang kunci 128-bit dipilih untuk menyederhanakan persoalan. Mode operasi ECB, walaupun tidak aman, akan digunakan untuk mendemonstrasikan algoritma AES dalam bentuk paralel.

II. ADVANCED ENCRYPTION STANDARD

Advanced Encryption Standard (AES) merupakan implementasi sebuah *cipher* blok simetris yang dapat memproses blok data berukuran 128 bit dan menggunakan kunci dengan panjang 128, 192, dan 256 bit berdasarkan algoritma Rijndael. Algoritma Rijndael didesain untuk dapat menangani ukuran blok data dan panjang kunci yang lain namun hal tersebut tidak diadopsi dalam AES.



Gambar 1 Algoritma AES

Fokus makalah ini adalah pada struktur AES keseluruhan seperti yang terlihat pada gambar 1. Algoritma AES menerima input sebuah blok dengan ukuran 16-byte yang digambarkan dalam bentuk matriks

persegi. Blok ini kemudian disalin ke dalam matriks *state* yang akan berubah di setiap tahap (*round*) enkripsi. Sembilan tahap pertama disusun oleh langkah-langkah yang sama yaitu *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundKey*. Pada tahap yang terakhir tidak ada operasi *MixColumns*. Proses *KeyExpansion* menerima 16-byte kunci dan menghasilkan array linier sepanjang 176-byte yang akan digunakan untuk operasi *AddRoundKey* pada tiap tahapnya. Pseudocode untuk algoritma AES adalah sebagai berikut.

```
Rijndael(State, CipherKey) {
  KeyExpansion(CipherKey, ExpandedKey);
  AddRoundKey(State, ExpandedKey[0]);
  for(i = 1; i < Nr; i++)
    Round(State, ExpandedKey[i]);
  FinalRound(State, ExpandedKey[Nr]);
}

Round(State, ExpandedKey[i]) {
  SubSytes(State);
  ShiftRows(State);
  MixColumns(State);
  AddRoundKey(State, ExpandedKey[i]);
}

FinalRound(State, ExpandedKey[Nr])) {
  SubBytes(State);
  ShiftRows(State);
  AddRoundKey(State, ExpandedKey[Nr]);
}
```

Rijndael telah mempublikasikan berbagai bentuk dan optimasi implementasi algoritma AES mulai pada prosesor 8-bit sampai 32-bit [1]. Pada prosesor 32-bit, AES dapat diimplementasikan dengan sejumlah operasi XOR, *table lookup*, dan *bit shift*.

SubBytes	$b_{i,j} = S[a_{i,j}]$
ShiftRows	$\begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix} = \begin{bmatrix} b_{0,j} \\ b_{1,j-1} \\ b_{2,j-2} \\ b_{3,j-3} \end{bmatrix}$
MixColumns	$\begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix}$
AddRondKeys	$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$

Tabel 1Persamaan operasi pada AES

Keempat transformasi dalam sebuah tahap dapat dinyatakan dalam bentuk aljabar seperti pada tabel 1, di mana $a_{i,j}$ adalah elemen matriks *state* dan $k_{i,j}$ adalah elemen matriks *RoundKey*. Pada persamaan *ShiftRows*, indeks kolom adalah dalam modulus 4. Keempat operasi transformasi tersebut dapat digabungkan dalam bentuk berikut,

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} S[a_{0,j}] \\ S[a_{1,j-1}] \\ S[a_{2,j-2}] \\ S[a_{3,j-3}] \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$
$$= \left(\begin{bmatrix} 2 \\ 1 \\ 1 \\ 3 \end{bmatrix} \bullet S[a_{0,j}] \right) \oplus \left(\begin{bmatrix} 3 \\ 2 \\ 1 \\ 1 \end{bmatrix} \bullet S[a_{1,j-1}] \right) \quad \textbf{(1)}$$
$$\oplus \left(\begin{bmatrix} 1 \\ 3 \\ 2 \\ 1 \end{bmatrix} \bullet S[a_{2,j-2}] \right) \oplus \left(\begin{bmatrix} 1 \\ 1 \\ 3 \\ 2 \end{bmatrix} \bullet S[a_{3,j-3}] \right) \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

dengan *S* adalah table S-box dan ● merupakan operasi untuk perkalian dalam *finite field* GF(2^8). Empat *lookup table* T_0, T_1, T_2, dan T_3 memuat hasil agregasi operasi pada persamaan (1) dan didefinisikan sebagai berikut.

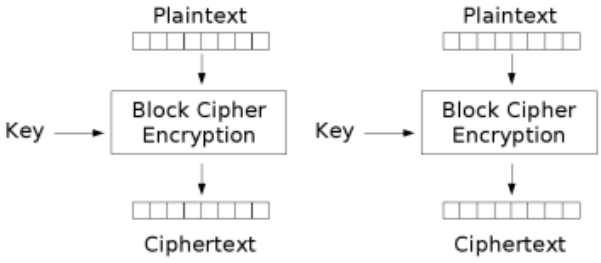
$$T_0[a_{i,j}] = \begin{bmatrix} S[a_{i,j}] \bullet 2 \\ S[a_{i,j}] \bullet 1 \\ S[a_{i,j}] \bullet 1 \\ S[a_{i,j}] \bullet 3 \end{bmatrix} \quad T_1[a_{i,j}] = \begin{bmatrix} S[a_{i,j}] \bullet 3 \\ S[a_{i,j}] \bullet 2 \\ S[a_{i,j}] \bullet 1 \\ S[a_{i,j}] \bullet 1 \end{bmatrix}$$
$$T_2[a_{i,j}] = \begin{bmatrix} S[a_{i,j}] \bullet 1 \\ S[a_{i,j}] \bullet 3 \\ S[a_{i,j}] \bullet 2 \\ S[a_{i,j}] \bullet 1 \end{bmatrix} \quad T_3[a_{i,j}] = \begin{bmatrix} S[a_{i,j}] \bullet 1 \\ S[a_{i,j}] \bullet 1 \\ S[a_{i,j}] \bullet 3 \\ S[a_{i,j}] \bullet 2 \end{bmatrix}$$

Karena $a_{i,j}$ adalah variable 8-bit, maka ada 256 macam nilai yang berbeda untuk $a_{i,j}$ sehingga ukuran untuk setiap tabel T adalah 1 KB. Untuk empat tabel T dibutuhkan 4 KB. Berdasarkan pada *lookup table*, fungsi dalam setiap tahap dapat didefinisikan sebagai berikut,

$$\begin{bmatrix} a'_{0,j} \\ a'_{1,j} \\ a'_{2,j} \\ a'_{3,j} \end{bmatrix} = T_0[a_{0,j}] \oplus T_1[a_{1,j-1}] \oplus T_2[a_{2,j-2}] \quad \textbf{(2)}$$
$$\oplus T_3[a_{3,j-3}] \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

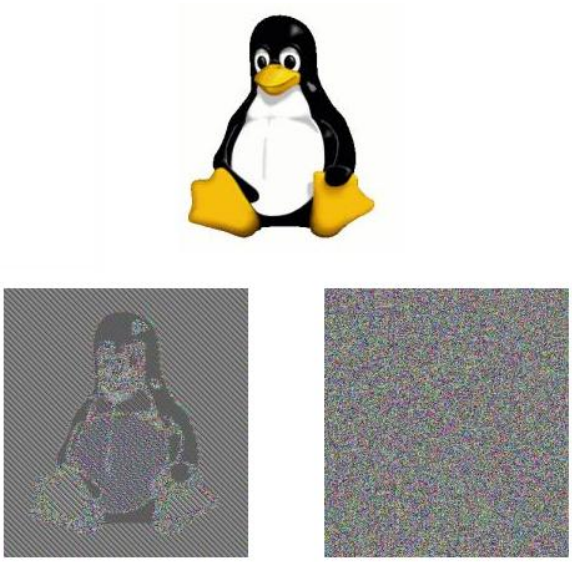
dengan $1 \leq j \leq 4$.

III. Mode Operasi Block Cipher



Gambar 2 Mode Operasi ECB

Cipher blok seperti AES dapat dioperasikan dalam berbagai mode operasi. Beberapa contoh mode operasi adalah *Electronic Code Book* (ECB), *Cipher-block Chaining* (CBC), *Cipher Feedback* (CFB), *Output Feedback* (OFB), dan *Counter* (CTR).



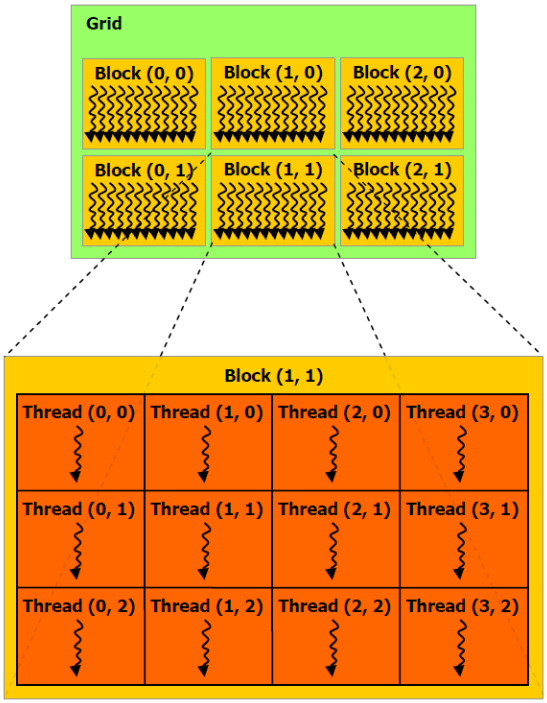
Gambar 3 Gambar asli, mode ECB, dan CBC [7]

Pada makalah ini dipilih mode operasi (ECB). Pada mode ECB, setiap blok plainteks yang sama akan selalu ditransformasikan menjadi cipherteks yang sama pula. Kelemahan ini dapat ditunjukkan pada gambar 4. Skema mode operasi ini dapat digambarkan seperti pada gambar 3. Blok-blok pada mode ECB tidak saling berhubungan. Tidak seperti mode lainnya seperti CBC atau CFB, setiap blok dihubungkan dengan blok lain, sehingga pola perulangan tidak terlihat pada cipherteks. Mode seperti CBC atau CFB tidak dapat diparalelisasi karena proses setiap blok harus menunggu proses blok sebelumnya selesai dijalankan. Walaupun mode ECB tidak aman, namun cukup untuk mendemonstrasikan aspek paralel pada AES.

IV. GRAPHICS PROCESSING UNIT DAN CUDA

Compute Unified Device Architecture (CUDA) adalah framework komputasi paralel yang dikembangkan oleh NVIDIA. Bahasa CUDA C adalah perluasan dari bahasa C yang mengijinkan pemrogram mendefinisikan fungsi bahasa C yang disebut *kernel*. *Kernel* ini, jika dipanggil, akan dieksekusi sebanyak N kali secara paralel oleh sejumlah N *thread* CUDA. Hal ini berbeda dengan fungsi C biasa yang yang dieksekusi sekali saja.

Setiap *thread* yang mengeksekusi *kernel* memiliki *thread ID* yang unik yang dapat diakses dari kernel melalui variabel **threadIdx**. threadIdx adalah sebuah vektor dengan tiga komponen. Dengan demikian *thread* dapat diidentifikasi menggunakan indeks *thread* satu, dua, atau tiga dimensi sehingga membentuk blok *thread* satu, dua, atau tiga dimensi. Hal ini memberikan cara yang natural untuk melakukan komputasi komputasi untuk elemen-elemen pada domain vektor, matriks, atau volume.



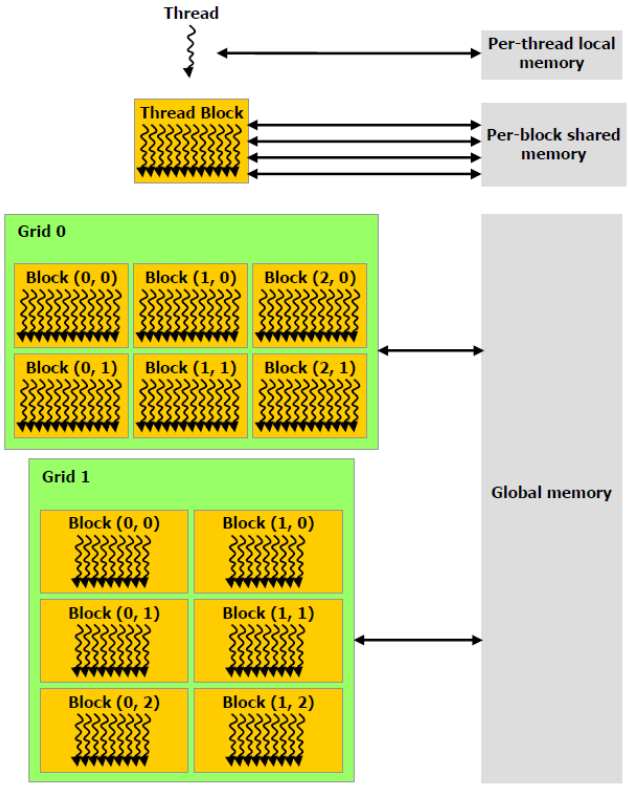
Gambar 4 Blok dan Thread [4]

Indeks sebuah *thread* dan *thread ID* berkaitan satu sama lain secara langsung. Untuk blok satu dimensi, *thread* dan *thread ID* adalah sama. Untuk blok dua dimensi dengan ukuran (D_x, D_y), *thread ID* untuk *thread* dengan indeks (x, y) adalah $(x+yD_x)$. Untuk blok tiga dimensi dengan ukuran (D_x, D_y, D_z), *thread ID* untuk *thread* dengan indeks (x, y, z) adalah $(x+yD_x+zD_xD_y)$.

Blok-blok diorganisasi ke dalam sebuah blok *grid of thread* satu atau dua dimensi, sebagaimana diilustrasikan pada gambar 5. Jumlah blok *thread* dalam sebuah *grid* biasanya ditentukan oleh ukuran data yang akan diproses atau jumlah prosesor yang ada pada sistem.

Thread pada sebuah blok dapat bekerja sama dengan berbagi data melalui *shared memory* dan dengan mensinkronisasikan eksekusi *thread-thread* untuk mengkoordinasikan pengaksesan memori. Lebih tepatnya, sebuah titik sinkronisasi pada kernel dapat dispesifikasikan dengan memanggil fungsi intrinsik **__syncthreads()**. Fungsi ini bertindak sebagai penghalang sehingga semua *thread* dalam sebuah blok harus menunggu sebelum diperbolehkan untuk melanjutkan eksekusinya.

Thread CUDA dapat mengakses data dari berbagai *memory space* selama masa eksekusinya, sebagaimana diilustrasikan pada gambar 6. Setiap *thread* masing-masing memiliki memori lokal. Setiap blok *thread* memiliki *shared memory* yang dapat diakses oleh *thread-thread* dalam blok tersebut. Semua *thread* mendapat akses ke memori global yang sama.



Gambar 5 Hirarki Memori [4]

Pada makalah ini, GPU yang digunakan untuk mengeksekusi kernel AES adalah NVIDIA GeForce GTX 470. GPU ini memiliki spesifikasi sebagai berikut.

- 14 Multiprocessor
- 1,22 GHz core clock
- 1280 MB main memory
- 64 KB constant memory
- 48 KB shared memory
- 1024 maximum thread per block

IV. Desain Paralel pada AES

Pada makalah ini, AES-ECB 128-bit dipilih untuk paralelisasi dan implementasinya berdasar pada bagian II makalah ini. Pada setiap tahapan enkripsi, algoritma AES bekerja pada matriks state berukuran 16-byte. Output matriks state pada satu tahap menjadi input untuk tahap berikutnya. Karena batasan ini, paralelisasi hanya dapat dilakukan pada setiap tahap (*round*) saja. Satu matriks state dapat ditangani secara konkuren oleh paling banyak 16 thread untuk melakukan *table lookup* dan operasi XOR.

Proses pembangkitan kunci adalah proses yang sifatnya sekuensial sehingga proses ini dilakukan oleh CPU. Kunci hasil proses pembangkitan ini kemudian ditransfer ke GPU. Proses pembangkitan kunci ini hanya dilakukan sekali di awal.

Layout operasi dan langkah-langkah yang dilakukan adalah sebagai berikut.

1. Pembangkitan kunci — Host code
2. Transfer kunci ke GPU — Host code
3. Transfer plaintext ke GPU — Host code
4. Jalankan kernel enkripsi AES — Device code
5. Transfer ciphertext dari GPU — Host code

Pertama-tama, CPU menghitung dan membangkitkan sekumpulan *round key* dari key yang diberikan. *Round key* yang dibangkitkan dan data plainteks kemudian ditransfer ke GPU. *Lookup table* tidak perlu ditransfer karena sejak awal sudah didefinisikan di *constant memory*. Setelah semua data yang diperlukan ditransfer ke GPU, kernel enkripsi AES dijalankan untuk seluruh plainteks. Setelah kernel selesai dijalankan, data cipherteks hasil perhitungan kernel ditransfer kembali ke memori di sisi CPU.

Implementasi paralel pada makalah ini adalah implementasi sederhana dan naif berdasarkan kode AES pada CPU yang dilakukan oleh [6]. Karena algoritma yang digunakan bekerja pada besaran 32-bit, maka matriks state yang berukuran 128-bit dipecah menjadi empat bagian 32-bit. Dengan demikian, setiap state ditangani oleh 4 thread secara konkuren. Pada makalah ini, jumlah thread yang digunakan pada setiap *block of threads* adalah 256.

Pada setiap *block of threads*, digunakan *shared memory* untuk memyimpan data state yang akan diproses oleh thread. Pemindahan data ke *shared memory* dilakukan karena akses data ke *shared memory* oleh thread lebih cepat daripada akses ke *global memory*.

V. Hasil Pengujian

Pada bagian ini akan disajikan perbedaan performa enkripsi algoritma AES pada CPU dan GPU. Hasil pengujian pada CPU didapatkan dari aplikasi OpenSSL. Di bawah ini adalah hasil pengujian dan perbandingan performa enkripsi pada CPU dan GPU. Spesifikasi CPU yang dipakai adalah Intel Core 2 Duo E7300 2.66GHz dengan memori 1 GB menggunakan sisten operasi Linux Fedora 12.

Chunk Size (bytes)	CPU Throughput (KB/s)	GPU Throughput (KB/s)
16	156452,75	87,77
64	173776,77	352,30
256	179107,93	1463,56
1024	180538,71	5510,49
2048	180288,85	8390,39
4096	180523,01	16290,89
8192	180704,60	32501,30
16384	180666,37	61162,08
32768	181002,10	108896,87
65536	180245,85	176616,04
131072	180826,76	267201,07
262144	180267,69	332557,37
524288	180355,07	396943,53

1048576	179755,89	393439,40
2097152	180355,07	399920,02
4194304	179755,89	405258,23
8388608	179361,68	411781,43

Angka-angka pada kolom GPU Throughput adalah hasil perhitungan jika waktu transfer data dari CPU ke GPU dan GPU ke CPU diperhitungkan. Berikut ini adalah tabel hasil pengujian jika waktu transfer tidak diperhitungkan.

Chunk Size (bytes)	GPU Throughput (KB/s)
16	347,28
64	1262,52
256	4743,47
1024	18859,38
2048	36400,70
4096	73185,01
8192	148809,52
16384	291375,29
32768	626959,25
65536	1591089,90
131072	2787456,45
262144	3851709,20
524288	4363239,71
1048576	4467401,93
2097152	4732677,66
4194304	4820729,14
8388608	4869419,66

Berdasarkan hasil pengujian di atas, implementasi algoritma AES pada GPU dapat memberikan percepatan sebesar 2,29 kali dibandingkan dengan CPU jika waktu transfer memori diperhitungkan. Throughput yang didapat adalah 411781,43 KB/s. Jika waktu transfer memori tidak diperhitungkan, percepatan yang diberikan adalah sebesar 27,15 kali dengan throughput 4869419,66 KB/s. Dari tabel di atas juga dapat diketahui bahwa algoritma AES pada GPU hanya efektif untuk data yang berukuran besar.

VII. KESIMPULAN

Implementasi AES pada GPU di makalah ini telah dapat memberikan percepatan sebanyak 2,29 kali (jika waktu transfer memori diperhitungkan) atau sebanyak 27,15 kali (jika waktu transfer memori tidak diperhitungkan). Implementasi yang ada di makalah ini adalah implementasi yang sangat sederhana dari kode AES untuk CPU yang sudah ada. Pengembangan algoritma AES untuk GPU seharusnya dapat ditingkatkan lebih lanjut dengan memanfaatkan dan mengeksploitasi fitur yang diberikan CUDA secara lebih baik, seperti *shared memory* dan *texture memory*.

REFERENSI

[1] Joan Daemen dan Rijmen, Vincent. The Design of Rijndael: The Advanced Encrpytion Standard. Springer. 2001.

[2] Chonglei Mei, Hai Jiang, Jeff Jenness, *CUDA-based AES Parallelization with Fine-Tuned GPU Memory Utilization*. 2009.

[3] NVIDIA Corporation. NVIDIA CUDA Architecture. 2010.

[4] NVIDIA Corporation. NVIDIA CUDA C Programming Guide. 2010.

[5] National Institute of Standard and Technology. Federal Information Processing Standards Publication 197, Announcing the Advanced Encryption Standard (AES). 2001.

[6] Philip J. Erdelsky. Rijndael Encryption Algorithm. http://www.efgh.com/software/rijndael.htm. Diakses 21 Maret 2011 pukul 20.00.

[7] Block Cipher Modes of Operation http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation. Diakses 22 Maret 2011 pukul 17.00.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 23 Maret 2011

Adityo Jiwandono, 13507015