AES Encryption and Decryption on a GPU

ARNAV KUMAR, 2016017 - Indraprastha Institute of Information Technology, Delhi
AAMIR TUFAIL AHMED, 2016001 - Indraprastha Institute of Information Technology, Delhi

The GPU computing project mid-deliverable report.

Additional Key Words and Phrases: cryptography, advanced encryption standard, graphics processing unit, security

ACM Reference Format:
Arnav Kumar and Aamir Tufail Ahmed. 2019. AES Encryption and Decryption on a GPU. 1, 1 (April 2019), 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 WHAT IS AES?

AES is the **Advanced Encryption Standard** which is also called by its original name “`Rijndael`” (pronounced as Rain-Dahl) is a method of encryption of data (electronic) by the U.S National Institute of Standards and Technology in 2001. It is a symmetric key block-cipher which supports blocks of sizes 128-bit.
The encryption method in AES uses a key which converts the plaintext data into a ciphertext (unreadable) and the decryption method in AES converts the same ciphertext into the original plaintext data. It is called a symmetric key cryptographic algorithm as the key used for both the encryption and decryption steps remain the same.

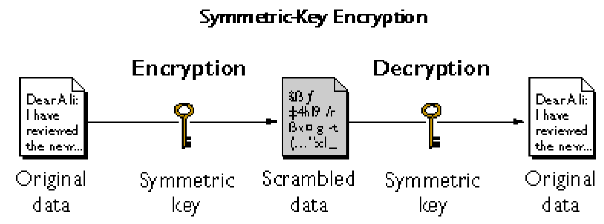


Fig. 1. Basis of symmetric key encryption, via IBM knowledge centre. (https://www.ibm.com/support/knowledgecenter/en/SSB23S_1.1.0.14/gtps7/s7symm.html).

Authors’ addresses: Arnav Kumar, 2016017 - Indraprastha Institute of Information Technology, Delhi, arnav16017@iiitd.ac.in; Aamir Tufail Ahmed, 2016001 - Indraprastha Institute of Information Technology, Delhi, aamir16001@iiitd.ac.in.

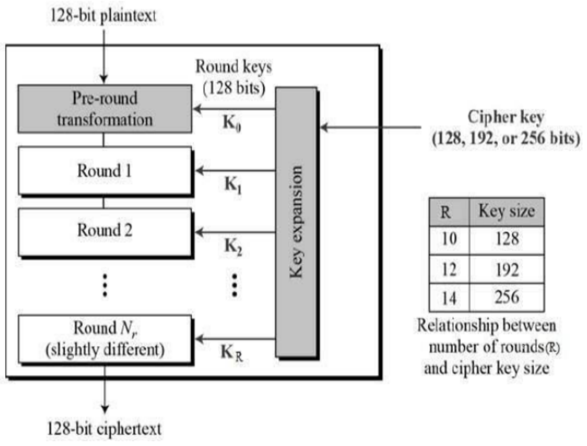
2 LITERATURE SURVEY

One of the good source of information about parallelizing AES is given in GPU Gems[1], which gives a low-level abstract implementation (using assembly and vertex program) of AES but that is a low-level modular assembly code (which is sequential on its own) although the article does give performance metrics if the code is written for a GPU and run on it.
This report mostly focuses on the sequential implementation of AES and its metrics so the content is about that. The original Rijndael whitepaper (AES Proposal: Rijndael)[2] was also a great help in understanding AES and its internal working.

3 ANALYSIS OF AES - HOW IT WORKS?

AES rather than being a Feistel cipher, is an iterative one. It's based on a 'substitution-permutation network' which is a series of operations that are linked and either involve replacing inputs by specific outputs (substitution) or involves shuffling bits around (permutations).
AES performs its computations on bytes rather than bits, therefore the plaintext block is treated as 16 bytes instead of 128 bits (1 byte = 8 bits). These 16 bytes are arranged as a matrix (4 columns and 4 rows) for processing.

For a 128-bit key, AES uses 10 rounds (the rounds differ as key length differs). Each and every one of these 'rounds' uses a different 128-bit round key, which is derived from the original AES key (using a key scheduling algorithm).



R	Key size
10	128
12	192
14	256

Fig. 2. Key expansion in AES (https://www.irjet.net/archives/V3/i8/IRJET-V3I8214.pdf).

3.1 Encryption Process

Each round of AES comprises of 4 sub-processes.

3.1.1 SubBytes (Byte Substitution). The input 16 bytes are substituted by looking up a table of fixed values called a S-Box which is pre defined. The result is a 4x4 matrix.

3.1.2 ShiftRows. Each of the 4 rows of the matrix are left shifted. Any entries that go out of bounds are looped back (re-inserted) on the right side of the row. The shift is done as-

- First row is not shifted.
- Second row is shifted by one byte to the left.
- Third row is shifted by two bytes to the left.
- Fourth row is shifted by three bytes to the left.

The result is a new matrix which consists of the initial 16 bytes but have been shifted wrt. each other.

3.1.3 MixColumns. Now each column of 4 bytes is transformed using a mathematical function(The details of the function are a little out of scope for this report). The function takes as input 4 bytes of one column and gives 4 completely new bytes which then replace the original column.
In the end, a new matrix is formulated which consists of 16 new bytes. (This step of MixColumns is not performed in the last round (10th round for 128-bit key)).

3.1.4 AddRoundKey. The 16 bytes of the matrix are now considered as 128 bits which are then XOR'ed to the 128 bits of the round key. In the last round, the output of this is the ciphertext. In all other rounds, the resulting 128 bits are used to begin another similar round of 16 bytes.

3.2 Decryption Process

The process of decryption of an AES ciphertext is same as the encryption process but in reverse order. In each round the 4 processes are performed in the reverse order i.e. AddRoundKey, MixColumns, ShiftRows and then Byte Substitution.

4 SCOPE OF PARALLELISM

The scope of parallelizing AES depends on which AES mode is being considered. There are 3 main modes of AES

- ECB(Electronic Code Book) mode
- CBC(Cipher Block Chaining) mode
- Counter mode

In CBC mode, the encryption of all blocks > 1 are dependent on the output of the (n-1)th block therefore parallelizing the encryption process of CBC AES mode is not possible. Although the decryption of CBC AES can be done in parallel.

In AES-ECB mode, the plaintext is divided into chunks of 16 bytes and then those 16 bytes each go through the AES mode. This mode will be the main focus of our GPU project as there is a huge scope of parallelism in this mode. Although this mode is not as secure as CBC mode as patterns in the ciphertext can be figured out in this mode.

In AES-Ctr (Counter) mode, each of the 16 bytes go through the rounds individually and a counter is used in addition with the key to add a level of randomness to the output. This mode is also in the scope of being parallelized and is almost as secure as AES-ECB mode.

5 SEQUENTIAL IMPLEMENTATION

The following are the results of the code written. The key size used is 16 bytes. The mode used is AES-128-ECB.

Metrics	
Code Used	Time Taken
Custom AES-ECB - 16 bytes	approx. 0
Custom AES-ECB - 160000 bytes	0.031250s
Custom AES-ECB - 1600000 bytes	0.234375s

6 NEXT SET OF MILESTONES

- Parallel implementation of encryption and decryption of AES for atleast one mode (building on top of the sequential code submitted)
- Analysis and comparisons with the existing sequential implementations of AES (OpenSSL, current code)
- Analysis of the methodology used for parallelizing the algorithm and how scalable it is.

7 REFERENCES

1: https://developer.nvidia.com/gpugems/GPUGems3/gpugems3_ch36.html
2: https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf
3: https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture8.pdf