Heterogeneous Cryptography: AES in Counter Mode

Clay Freeman
Computer Science, Arkansas State University, Jonesboro, Arkansas, United States of America

Abstract – *Privacy is a growing concern for many people in society. Many manufacturers of computers and peripherals are starting to provide options for encrypting data to meet this demand for greater privacy. However, fast cryptography is needed to offset the tradeoff between security and performance. The Advanced Encryption Standard (AES) provides a basis for cryptography that can be used to conceal data at rest or in transit. AES, when used in counter mode (CTR-mode), is capable of parallelization at high speeds, whether in hardware or simulated by software. This paper will discuss how AES-CTR can be used for such high-throughput scenarios.*

Keywords: Advanced Encryption Standard, Counter, Mode of Operation, AES, CTR, Heterogeneous, Cryptography

1 Introduction

The Advanced Encryption Standard, also known as Rijndael, is an encryption and decryption algorithm that supports 128-bit input blocks with key sizes of 128-bits, 192-bits, and 256-bits [1].

AES in its 128-bit key variant is used heavily throughout this paper. This paper will describe the implementation details and performance characteristics of AES in Counter Mode across a heterogeneous array of devices.

2 Background

AES in Counter Mode (AES-CTR) is a recommended mode of operation for AES (among others; e.g. ECB) [2] that was chosen as a candidate for this paper due to its highly suitable nature for parallel encryption and decryption and its capability to support random read-write access. Because of how AES-CTR works it only requires the encryption portion of the AES standard to generate a key stream. Encryption and decryption are reversible actions that occur when the plain text or cipher text are XOR'ed with the key stream.

AES-CTR works by encrypting a 128-bit state block that is half a random nonce value and half an integer counter value with a key of 128, 192, or 256 bits in size. The counter value can be arbitrary; usually it is some sort of predictable, incremental constant available during encryption. For this paper's implementation (and most other implementations), the current block index is used as the counter value.

2.1 Assurances and Disclaimers

It is important to note that the counter mode of block ciphers uses a "one-time pad" exclusive-or technique, a very important consequence of which is that keys cannot be reused: plain text can be recovered from two or more cipher texts with the same key stream. It is imperative that proper care is taken when using this mode of operation for AES to ensure that key streams are unique throughout the lifetime of all generated cipher texts.

3 Principles of Information Security

As a conceptual aside, let's look at some of the fundamental properties of information security. The ISO/IEC 27000:2014 publication defines *information security* [3] as the "preservation of *confidentiality* (2.12), *integrity* (2.40) and *availability* (2.9) of information," and states that "In addition, other properties, such as *authenticity* (2.8), accountability, *non-repudiation* (2.54), and *reliability* (2.62) can also be involved." This publication goes on to define these terms, but they will be left out of this paper for brevity.

3.1 Relevance to AES-CTR

Of the previously defined properties, AES-CTR only exhibits the property of *confidentiality*. Other properties such as *integrity, authenticity*, *non-repudiation*, and *reliability* are not inherently built-in to any mode of operation for block ciphers (some modes facilitate such properties) but can be added through other means. A strong cryptosystem should exhibit most, if not all, of these properties depending on the situation.

4 Algorithm Details

Let's begin by selecting possible candidate block cipher modes of operation for parallel cryptography. The purpose of this paper is to explore how parallelism can aid in improving the throughput of cryptography, however, most of the block cipher modes of operation are designed in a way that do not support parallelism.

The two possible candidates for parallel cryptography that this paper will examine are the Electronic Codebook (ECB) and Counter (CTR) modes. Other common modes do not support parallel encryption, decryption, and random read-write access.

4.1 Why Counter Mode versus Electronic Codebook?

Figure 1 — Original "Example" Text

In the simplest block cipher mode of operation, ECB, each plain text block is encrypted one at a time using an identical key and is then replaced with the resulting cipher text [2]. The image in Figure 1 (above) contains the black text "Example" over a white background. Figure 2 (below) shows the image from Figure 1 encrypted using ECB mode.

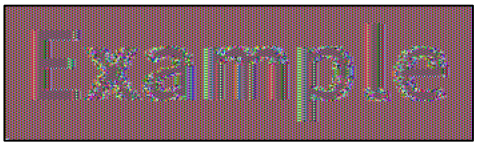

Figure 2 — Encrypted "Example" in ECB Mode

As can be seen in Figure 2, the pattern of "Example" can still be made out even though the input has been encrypted. This is because repeated plain text inputs result in identical cipher text outputs. Figure 3 (below) shows the image from Figure 1 encrypted using CTR mode.

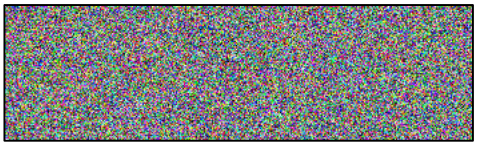

Figure 3 — Encrypted "Example" in CTR Mode

In Figure 3, the "Example" text is completely and randomly concealed. No patterns emerge representing the original text. This is because the key stream generated from the counter mode of operation is completely unique for each block.

4.2 The Discrete Steps of AES Encryption

AES encryption operates on a 128-bit state matrix with the help of distinct intermediate keys for each round. The process by which each round key is generated is performed in the algorithm's set-up stage.

Furthermore, AES has multiple additional round-steps during which encryption takes place. Each step is described below:

1. ***KeyExpansion***: AES uses independent keys for each round of the algorithm. The key expansion stage generates the required keys for each round from the input key [1].

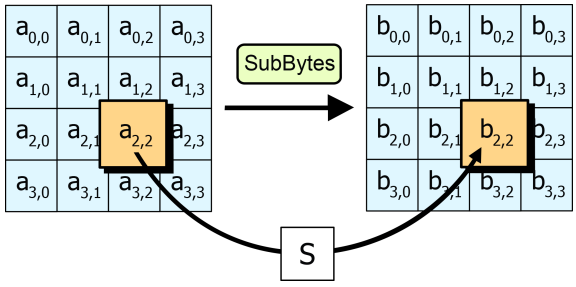

Figure 4 — ***SubBytes***

2. ***SubBytes***: Each byte of the state is replaced using an "S-box" lookup table which contains the modular multiplicative inverse values for every possible byte value over GF(2^8) [1].

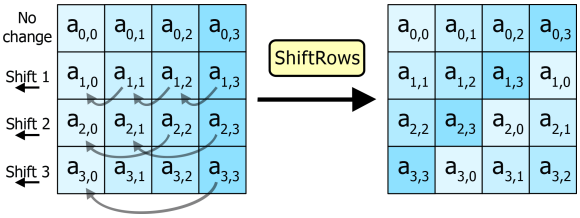

Figure 5 — ***ShiftRows***

3. ***ShiftRows***: A cyclic left-shift of each row is performed. The shift amount of each row is dependent on its zero-indexed row number [1].

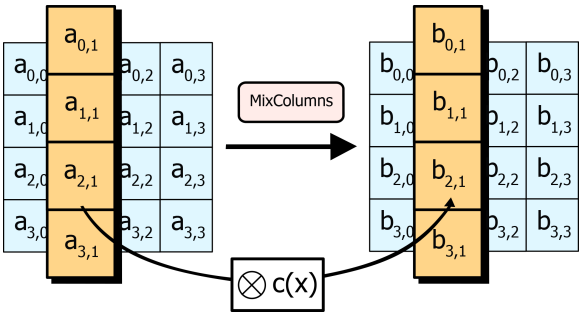

Figure 6 — ***MixColumns***

4. ***MixColumns***: Each column of the matrix is multiplied by a constant matrix over GF(2^8) mod $x^4 + 1$ [1].

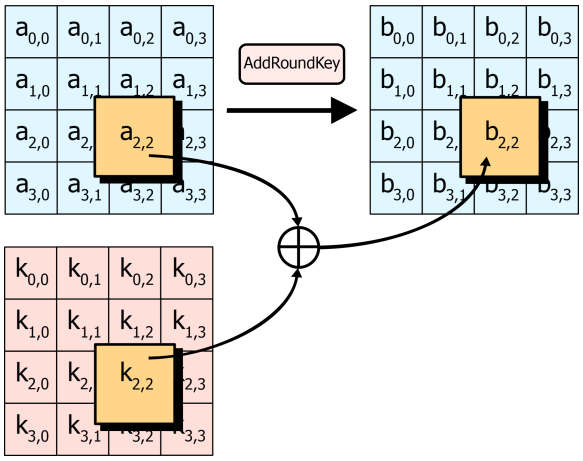

Figure 7 — *AddRoundKey*

5. *AddRoundKey*: Each byte of the state is XOR'ed with the corresponding byte of the intermediate key for that specific round [1].

4.3 Algorithm Overview

The first round of AES encryption is always just simply *AddRoundKey* on the state matrix using the input key. The middle rounds of AES encryption consist of *SubBytes*, *ShiftRows*, *MixColumns*, then *AddRoundKey*. The last round of AES encryption consists of *SubBytes*, *ShiftRows*, then *AddRoundKey* [1].

MixColumns is omitted from the last round of the algorithm because the complexity it adds is negligible such that it has no benefit to security and would only serve to reduce throughput of the algorithm [4].

5 Methodology

This paper's implementation of AES128 CTR uses the Open Computing Language (OpenCL). Each kernel was responsible for encrypting or decrypting one 16-byte block of data. Up to 1 GiB of data (over 67 million kernels) could be linearly queued at any given time on the target device.

Each test was fine-tuned using three parameters to gauge how the implementation would react in a variety of situations. The first parameter was file size. A wide gamut of sizes was tested from 1 MiB – 1 GiB to differentiate between short-lived and sustained performance. The second parameter was the buffer size limit. This parameter ultimately decided the maximum number of kernels that would be queued at any given time. Buffer sizes from 16 KiB to 1 GiB were tested. The last parameter determined which OpenCL device would be used to run the tests.

16 trials of each parameter configuration were recorded which consisted of 8 encryptions and 8 decryptions. The performance between encryption and decryption should be identical since AES-CTR is reversible.

All tests were performed on an AMD® Radeon™ Pro 460 Compute Engine with 4,096 threads @ 907 MHz and an Intel® Core™ i7-6920HQ CPU with 4 cores (8 threads) @ 2.90 GHz. The input being encrypted resided on a RAM disk and the host operating system was macOS 10.13.4.

5.1 Performance Characteristics

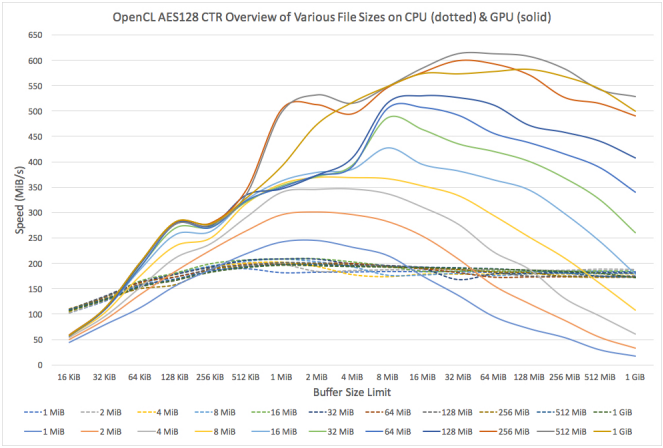

Figure 8 — Performance Overview

As can be seen right away the GPU greatly outperforms the CPU when dealing with large files (greater than 256 MiB). Figure 8 demonstrates that up to 3 times the performance can be expected from the GPU in this type of scenario.

The GPU seems to suffer with smaller buffer sizes, while the inverse is true of the CPU. Both devices seem to suffer from a performance wall after specific buffer sizes.

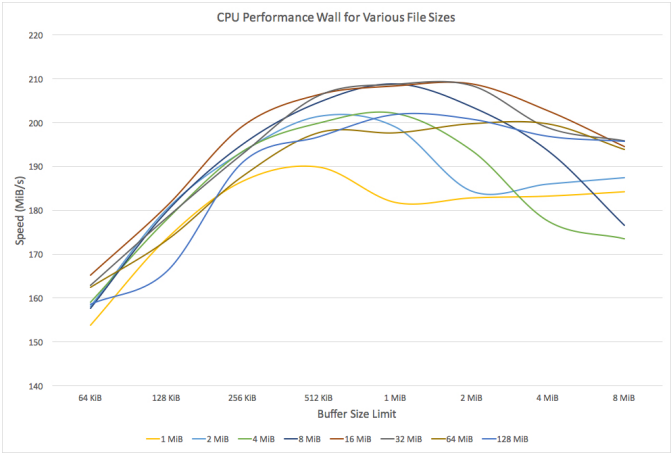

Figure 9 — CPU Performance Wall

Figure 9 demonstrates the performance barrier encountered on the CPU as soon as the buffer size is increased to 512 KiB. This performance plateau could be due to a potentially large number of context switches when running on the device, the locality of each kernel to a single 128-bit block, or a limitation of memory bandwidth. Further investigation may yield useful information for alleviating this bottleneck.

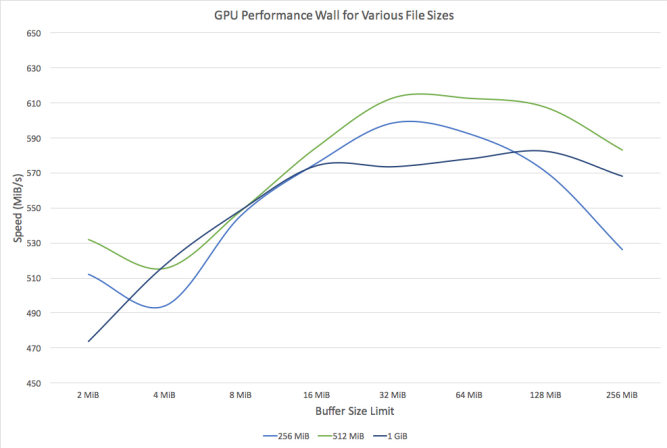

Figure 10 — GPU Performance Wall

The GPU encounters its performance wall much later with a buffer size of 32 MiB. This is likely due to bandwidth limitations of 935 MB/s on the PCI Express 3.0 bus [6], but could also be due to the locality of the kernel to a single 128-bit block. Kernels working on multiple blocks may not be adversely affected.

6 Applications

With a proper implementation, AES-CTR could be extremely useful for high throughput data transfers across a network. AES-CTR would need to be paired with a secure, ephemeral key exchange algorithm to ensure safety against the key reuse issue. Messages would also need to be authenticated to curb single-bit permutations and foul play.

7 Future Work

The OpenCL kernel implementation of this paper needs to be audited more carefully for potential hot spots that may hurt performance. Also, it may be advantageous for each kernel to be responsible for a set of 128-bit blocks instead of only one. This could greatly increase efficiency since a larger volume of work might be produced with fewer kernels.

The algorithm presented in this paper for AES-CTR is a full software implementation. Hardware implementations for AES encryption could be used for even higher performance. Intel's AES-NI extension to the x86 instruction set architecture can achieve sequential throughput of approximately 600 MiB per second [5]. This metric is very close to the performance of this paper's kernel on the GPU and has the potential to vastly outperform it with a parallel implementation.

8 Conclusion

AES-CTR, the counter mode of operation of the Advanced Encryption Standard, is a highly parallel and flexible cipher. The performance of AES-CTR seems to scale with buffer size until a plateau is reached.

Ideally, one should tune the execution of this algorithm to best suit their environment. In this paper's environment, the GPU performed best at a buffer size of 32 MiB and the CPU at a buffer size of 1-2 MiB. The performance measured at these buffer sizes should be sustained at larger file sizes.

Most importantly, AES-CTR seems to provide an acceptable tradeoff between high security and fast throughput for most applications, keeping other algorithms and modes of operation in mind.

AES-CTR has been an interesting algorithm to implement, and there is much room for innovation using this variant of the cipher.

9 Attribution Notice

Several figures included in this paper were found on Wikimedia Commons in the public domain, thus not requiring attribution. However, links to these figures are provided below:

Figure 4: https://en.wikipedia.org/wiki/File:AES-SubBytes.svg
Figure 5: https://en.wikipedia.org/wiki/File:AES-ShiftRows.svg
Figure 6: https://en.wikipedia.org/wiki/File:AES-MixColumns.svg
Figure 7: https://en.wikipedia.org/wiki/File:AES-AddRoundKey.svg

10 References

[1] "Announcing the ADVANCED ENCRYPTION STANDARD (AES)"; Federal Information Processing Standards Publication 197, November 2001.

[2] "Recommendation for Block Cipher Modes of Operation"; NIST Special Publication 800-38A, §6.1 (Pg. 9) and §6.5 (Pg. 15), December 2001.

[3] "Information technology — Security techniques — Information security management systems — Overview and vocabulary"; ISO/IEC 27000:2014, January 2014.

[4] "A Stick Figure Guide to the Advanced Encryption Standard (AES)"; Moserware, Act 3: Scene 15, September 2009.

[5] Grant McWilliams; "Hardware AES Showdown - VIA Padlock vs Intel AES-NI vs AMD Hexacore"; July 2011.

[6] Nathan Edwards; "Theoretical vs. Actual Bandwidth: PCI Express and Thunderbolt"; Tested, September 2013.