Journal of Theoretical and Applied Information Technology
15th August 2019. Vol.97. No 15
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

PERFORMANCE ENHANCEMENT OF THE ADVANCED ENCRYPTION STANDARD VIA PIPELINED IMPLEMENTATION

[1]MALEK M. BARHOUSH, [2]NAJIB A. KOFAHI, [3]KHALID M.O. NAHAR, [4]ANAS M.R. ALSOBEH, [5]AMEERA JARADAT, [6]BAYAN ALOMARI

[1, 2, 3, 5, 6] Department of Computer Sciences, Faculty of Information Technology & Computer Sciences, Yarmouk University, Irbid-Jordan

[4] Department of Computer Information Systems, Faculty of Information Technology & Computer Sciences, Yarmouk University, Irbid-Jordan

E-mail: [1]malek@yu.edu.jo, [2]nkofahi@yu.edu.jo, [3]khalids@yu.edu.jo, [4]anas.alsobeh@yu.edu.jo, [5]ameera@yu.edu.jo, [6]beboalomari12@gmail.com

ABSTRACT

Information and computer security become a key issue these days. This is due to rapid developments in data communications and computer technologies. Hence, there is a serious need for a secure transmission of both data and information between senders and receivers. Since there is no fully secured communication system for Web-based systems, communication systems rely heavily on complex and difficult cipher systems. Cipher systems usually consist of two main parts; Encryption and Decryption to hide and secure both transmitted data and information on long trunks. In addition, storing both important and sensitive data and information requires securing them from intruders, which can be handled by encrypting them. This paper studies one of the most important and widely used secret key encryption/decryption algorithms, namely the Advanced Encryption Standard (AES). The implementation of the AES algorithm involves complex computational steps that have made the implementation of these steps slow and time consuming. Using state of the art of multi-core architecture, we propose a pipelined implementation of the AES algorithm to reduce both computation complexity and elapsed computation time. Our contribution, the proposed AES implementation does not require dedicated equipment, it works with any kind of computers that are available to the public, such as Intel-based computers. A comparison of CPU performance is performed on both pipelined and sequential implementations on different file sizes. We have found that our pipelined implementation outperforms the sequential one, without the use of any special equipment. What distinguishes our work from the rest of the work done by several researchers is that the proposed AES implementation is designed for multi-core computers that do not have expensive equipment such as GPU or FPGA, such devices are widely available.

Keywords: *Cryptography, AES Encryption & Decryption, Pipelined Advanced Encryption Standard, Web-based application.*

1.0 INTRODUCTION

Security of data and information is a key issue in today's communication and network information systems, where data and information are exchanged electronically across the web. There is an urgent need to defend sensitive information against intentional attacks as it is moving through public communication or cyberspace [1] [2].

Encryption and decryption mechanisms are the easiest, cheapest, more effective and more flexible way to protect digital information within public networks [3]. The world of encryption facilitates the ability of Web-based systems to hide and retrieve information when needed, enabling users to maintain their personal information and communicate with others through secure channels [4]. Cryptographic technologies support secrecy service, which prevents unauthorized entity from accessing critical information. Secrecy service is achieved by replacing the original data with encrypted code. Cryptographic technologies also provide many services for the original data, such as integrity, authenticity, non-repudiation, privacy and digital signature [5] [6].

Cryptography is a technique that uses a specific ciphering algorithm to convert plaintext into unintelligent text that is hard to recover without obtaining the decryption key. Since data is exchanged widely through the public networks (The Internet), safe exchange of data through communication lines requires a strong and reliable encryption algorithm. Faulty or breakable cryptographic algorithm makes information interchange between parties risky, which in turn makes the communication system unreliable. Many factors are taken into consideration when choosing to implement a cryptographic algorithm such as, execution time, security level and computation power, memory required [4] [7] [8] [9] [10] [11] [12] [13] [14].

DES algorithm has become insufficient for securing newly emerging applications [15] [16]. A new version of the classical DES was formulated by extending the key size and applying three different keys to encrypt the data, which was referred to as a 3-DES (or TDES). Unfortunately, the performance of 3-DES was inefficient [17]. Advanced Encryption Standard (AES) is a modern ciphering standard used by several applications to secure sensitive data and information, as well as transfer them over insecure networks [12] [18] [19]. In 1997, the National Institute of Standards and Technology (NIST) conducted a race to formulate a new standard for encryption and decryption. As a result of the contest, AES was elected by NIST as a new standard technology for both encryption and decryption [19] [20] [21] [22] . AES was formulated by Rijndael, it is a subset of Rijndael family's algorithms, each with a different block and key sizes. AES uses a block size of 128 bits and three versions of keys: 128 bits, 192 bits and 256 bits [19] [20] [21]. AES is more secure and has a good performance and more flexibility for many hardware and applications [23].

In this research paper, we focused on the AES algorithm as a modern encryption standard, and our goal is to improve the speed of the algorithm implementation using the tools available in our hands. Since the modern machines are made up of more than one core, there is no need to use special equipment for acceleration. We use software pipelined implementation based on multithreading in JAVA programming language. The main contribution of this work provides an enhancement to the AES algorithm performance using software pipelining (task decomposition) and multicore architecture, without the need for special and costly hardware or software.

The rest of the paper is organized as follows: Section 2.0 presents previous work. After that, Section 3.0 introduces AES Sequential Algorithm. Section 4.0 presents hardware and software AES implementations. Section 0 describes our approach, while section 6.0 analyses pipelined AES implementation. The results is shown in section 7.0. Finally, Section 8.0 concludes the work.

2.0 RELATED WORK

In cryptographic terms, we call data and information that we can simply read without any distinctive treatment a plaintext. The routine of altering the original text in a way to hide the essence is called encryption. Encrypting plaintexts will result in vague and unreadable texts, and those are called ciphertext. Encryption is one way to hide readable data from ineligible users. Figure 1 illustrates the encryption and decryption Processes, where in the encryption process a message is converted to a ciphertext, while in decryption process, the original plaintext is extracted from the ciphertext [24]. Assuming that encryption and decryption are done between two parties, each must have a secret Key to encrypt the transmitted data or decrypt the received encrypted data [24].

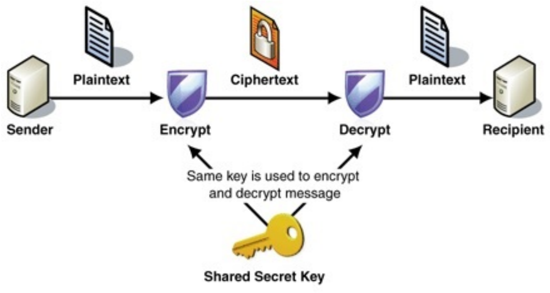


Figure 1. Ciphering Process [25].

2.1 Definition of Cryptography

Cryptography or Cryptology is related to the Ancient Greek. The word cryptography consists of two parts: the first, “crypto”, meaning secrecy and the second, “graphy”, meaning writing or logy which means “to speak”. It is known as the practice and study of hiding information. The word graphy appears in mathematics, which teaches mathematical techniques used to hide big numbers [26]. There are two trends in the world of cryptology: cryptography and cryptanalysis.

Cryptography is a two-way process: converting text into an incomprehensible text using a key, and vice versa. Cryptanalysis is the extraction of text from incomprehensible text without obtaining the decryption key in a legitimate manner [1]. Cryptography allows hiding sensitive information or data when either transferred in the risky communication system or stored for long range. As far as the cryptic algorithm is complicated and has a long secret key, it is hard to be breakable by any intruder. Recently, encryption is considered a branch of many disciplines, such as: mathematics, computer science, information theory, computer security and computer engineering. Cryptography is used in various applications such as ATM and smart cards, access control and internet security [27] [28].

The safety and confidentiality of cryptographic implementations lie in the computation time consumed in both encryption and decryption. Some complicated and high computation algorithms need hundreds or even thousands of years to decipher a message despite of today's available computing power [17].

2.2 The Purpose of Cryptography

The Egyptians were first to use the cryptography 1900 BC, where they were engraved in hieroglyphics [26]. In Web-based systems, the cryptography refers to the mechanism that enables individuals to achieve better security when storing data and information and when they are transmitted over communication lines. There are many techniques to achieve better security and require that this be reflected on encryption to achieve this goal.

Many researchers attribute the appearance of encryption sometime after the invention of writing, with a wide range of applications ranging from communication between diplomats and combat plans during wartime. They noticed that new forms of cryptography have appeared as a result of the widespread development of computer communications [26]. Therefore, there are many services provided by the world of cryptography when establishing a communication between two or more parties, including: confidentiality, authentication, privacy, integrity and non-repudiation [5] [6] [23].

Two main categories of cryptographic schemes to achieve these functions, namely: symmetric (secret key) and asymmetric (public-key) cryptography. In the next section, we will describe these techniques. In both cases, symmetric and asymmetric, the original text is called plaintext and the output of the encryption process is the ciphertext [6] [29].

2.3 Classification of Cipher Systems

Cipher systems were developed over time from classical systems that usually based on the substitution or transposition of the characters to modern cryptography, where modern cryptography techniques depend on the presence of a key. Figure 2 illustrates the hierarchical classification of the current cipher systems [17].

Modern cryptography systems are either symmetric or asymmetric ciphers. In symmetric cipher the decryption stage is the reverse of the encryption stage and the key used in both encryption and decryption processes are same. This indicates that both encryption and decryption processes have same time and space complexity. On the other hand, in asymmetric, public-key, cryptography, two different keys: public and private keys are used for encryption and decryption processes. The time required for encryption differs from the time of decryption. In general, public keys can be accessed by public people, public key is used for encryption when a secrecy service is needed, and the private key is in the hand of owner and used for decryption process [24] [26]. In our research we focus on a symmetric encryption, block cipher, namely AES.

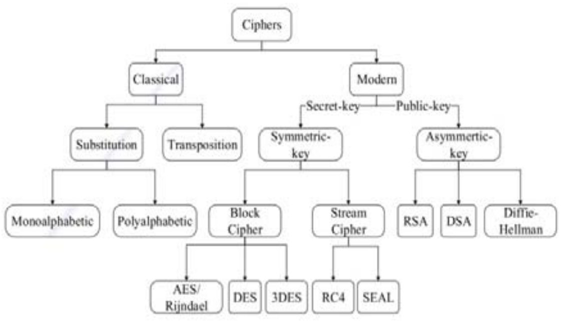


Figure 2. Cipher Systems Classification [17].

In 1977, IBM developed Data Encryption Standard (DES) algorithm, which was adopted by US government as an official standard. However, due to the new developments in technology, the DES is no longer secure in its standard form, while it is still useful in its updated form [12] [23] [30] [31].

In 1979, the DES was broken by a machine implemented by Diffie and Hellman. They declared that given a small piece of text and match cipher text, the key is found by the breakable machine in less than 24 hours, knowing that the search space for the key is 2^56 [32]. In 1979 the DES was replaced with 3DES, which was slower than any other block cipher algorithm [18] [33]. Then, the US Federal Register announced on September 12, 1997, NIST's desire to develop a new encryption standard [22] [34].

On November 26, 2001, the Federal Information Processing Standards Publication no 197 announced the new standard named Advanced Encryption Standard (AES). AES is used as a standard algorithm for encryption in 2001. AES becomes one of the most useful symmetric blocked cipher algorithms that support different key sizes [20] [34] [35].

3.0 AES SEQUANTIAL ALGORITHM

In AES standard, the input block size is 128 bits, the size of the output block is 128 bits, and the state size is 128 bits. 128 bits are denoted by four words, each of 32-bit or four bytes (Nb = 4). In other words, the state is arranged in four columns, each of four bytes. The key (K) size in AES either 128, 192 or 256 bits, which is given as Nk = 4, 6, and 8 of 32-bit words. In other words, key is represented by the number of columns in the cipher key, where each column is 32-bit words. The number of rounds (Nr) in AES algorithm depends on the length of cipher key, Nr equal to 10, 12 or 14 for Nk equal to 4, 6, and 8 respectively [20] [36].

AES is a cryptographic algorithm that encrypts data after dividing it into blocks of equal size, and then apply the algorithm and key to these blocks. The size of blocks is 128-bits and the size of key is variable. AES is an iterated symmetric-cryptographic algorithm that uses state of 16 bytes arranged in the form of two-dimensional bytes of (4 by 4) for 128 block size. The transformations in the algorithm are performed on the state. The input and output blocks as well as the key are treated as byte array of bytes [20] [36]. Figure 3 shows AES algorithm top level blocks, the figure clearly shows both the basic inputs and outputs of the algorithm.

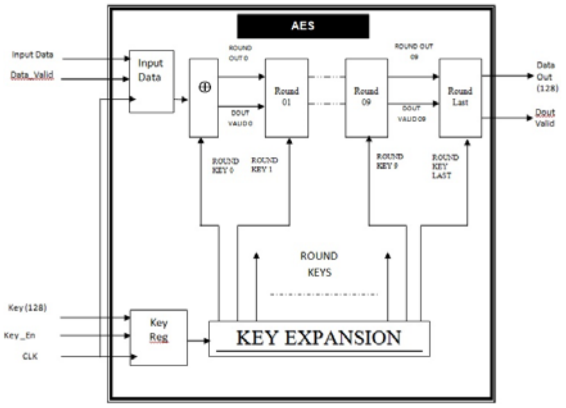


Figure 4. The AES Algorithm Top Level Block Diagram [37].

In standard AES, the last round of the 10 rounds for key size of 128 bits will be performed separately. The AES algorithm uses a round function consisting of four different transformations. These transformations are byte oriented. The same transformation functions are used for encryption and decryption. The transformation functions are [37]:

- ✓ Byte substitution operation, where the AES algorithm reacts with substitution table (S-box).
- ✓ Shift rows operation, where the algorithm reacts with the State's rows.
- ✓ Mix Columns operation, where the algorithm reacts with the State's columns.
- ✓ Add Round Key operation, where the algorithm performs the operation XOR bit wise with 128 bits data and sub key.

The four operations are performed for each single round, however the third operation is not performed in the last round, which requires the last round is carried out separately. Depending on key size used, a fresh set of keys is produced, these keys are fed to a corresponding round [20] [38] [39]. In order to examine the power of any cipher system, it should meet some factors like: Firm and strong security, Speed, Ease of implementation (simple mechanism), and Flexibility (use of different key length) [6]. The purpose of this research is to enhance the AES's performance using software pipelining (task decomposition) and multicore architecture.

There are many details concerning mathematical preliminaries of AES cryptography algorithm such as confusion, diffusion, key expansion, scheduling, S-box, Inverse S-box, addition and multiplication. There are a variety of examples and more details about AES encryption and decryption processes in [20] [36] [38] [39] [40] [41] [42] [43] [44].

4.0 HW/SW AES IMPLEMENTATIONS

Many researchers have tackled the problem of AES performance improvement, some of them went through hardware development while the others went through software improvement. In this research, we tackled the AES performance enhancement from software side through software pipelining. Next sub sections will briefly get into an overview of researchers' contribution in both sides: hardware and software.

4.1 HARDWARE (HW) ENHANCEMENT

In [8], the authors pointed out that the cost and performance of implementing any cryptographic mechanism should be feasible. The authors study the performance of the AES algorithm on a variety of common platforms: 32-bit CPUs, 64-bit CPUs, 8-bit smart-card CPUs, DSPs and hash functions in many hardware applications were presented. General observations on the performance issues for each of the platforms were taken into consideration. The authors recommended that AES need to be: efficient on the smaller, weaker 32-bit CPUs, it also needs to be parallelized, able to work on DSPs, and need to efficiently work as a hash function.

The structure of pipelined hardware architecture for AES RIJNDAEL were defined by [42], they used heuristic rules to choose best hardware implementation and deeply evaluate it. They use FPGAs/Xilinx Virtex-E technology for their implementation, they consider the FPGA's constraints in order to define an optimal pipelined technique that takes into consideration the AES place and route constraints. The authors used loop unrolling technique to improve pipeline outcome. They demonstrated that Sub Bytes Transformation can be parallelized. They were able to improve the ratio throughput (speed / slice) by 25% for the Xilinx Virtex1000. The throughput reaches 18.5 Gbits/sec with only 542 slices and ten blocks of RAM.

In 2010 an AES hardware architecture was designed and deployed on the top of FPGA and VHDL [45]. FPGA stands for a hardware called Field Programmable Gate Arrays [16], and VHDL stands for Very High Speed Integrated Circuit Hardware Description language. The authors use XCV600 of Xilinx Virtex Family and VHDL language to implement 128-AES. They tested and debugged their work using Xilinx ISE 8.1 software. The authors simulated their work using Xilinx xst. The authors claimed that the hardware implementation for 128-AES provides more security and more performance, so it is an urgent requirement for wireless communication and cellular systems. Their 128-AES design works on frequency of 140.390MHz, 1853 slices, 512 slice flip flops, 20 units of 256x8-bit ROM, and 130248 kilobyte of total memory usage. Both 128-bit encryption and decryption process implementation were programmed in VHDL. The throughput of encryption/decryption of their product reached up to 352 Mbit/Sec.

An optimized 128-AES hardware employing FPGA and VHDL was made by [46]. Their 128-AES hardware optimized multiplier architecture. The author claims that the result is cost effective and secure, it was deployed using Xilinx 14.2. In their design, they use less hardware resources with respect to different platforms. A code was written using VHDL for both 128-bit encryption and decryption processes. The result was tested via Xilinx 14.2., it is 6% space reduced on FPGA Spartan 3e.

Because AES is the best and most flexible and efficient cryptosystem used to secure data, another effort using an FPGA and VHDL to implement AES appears in [22]. The authors propose 128-AES hardware design to increase AES performance through FPGA and VHDL. They simulate their design via Xilinx ISE 12.3i. The result gives a throughput of 1609 Mbit/Sec.

The authors in [43] realized that the servers in cloud systems need AES accelerators in order to reduce the computation time for securing communications. The authors have put in place the necessary precautions to speed up the encryption and decryption process, they provide a memory with multiple ports, so that it allows reading and writing more than one location at the same time. They pipelined and unrolled the 128-AES algorithm, so at each clock cycle, there will be many activities. Their pipelined implementation is simulated and synthesized on XC7VX690T chip, their throughput is 104.06 Gbps at a frequency of 813 MHz, and the maximum efficiency was 30.74 Mbps.

4.2 SOFTWARE (SW) ENHANCEMENT

Secure Web service requires a high speed in dealing with security, and as these services consume heavy calculations, there is a need to accelerate these services. In [47], the authors focus on speeding up AES cryptosystem, they implemented AES using parallel computing platform called CUDA. The authors designed a parallel AES that run on NVIDIA GeForce 8 GPU, the code is implemented with CUDA. They used OpenSSL implementation to show their results. Normally a GPU contains hundreds of cores, each have many threads. CUDA is a programming model used to program NVIDIA GeForce GPU family. The implementation was optimized and improved, as well as, the throughput was increased 14 times compared with a faster CPU at that time. Using CUDA, different parallel implementations were spawned with collaboration with hardware control resources. The ultimate goal is to produce the best performance.

For improving the performance of secure web services, the author in [48] presented another attempt to parallelize AES implementation Based on the GPU and multiple thread of control. The GPU is designed for intensive graphical computations. The authors created many threads in AES process, they assign each thread to one GPU-processing-element. They divide the AES text data into blocks, then assign each block to one thread. The shared resources of parallel AES are stored in the global GPU space, so each AES thread can run in parallel along with its corresponding block. Results of [48] show that the enhancement of their AES over GPU was 7x faster than AES over the CPU.

Many applications, such as Web services, smart cards and digital multimedia services, use AES. Acceleration of the AES implementation has become an urgent requirement, so researchers in [49] investigated many parallel AES implementations in term of data-level & task-level parallelism, loop unrolling, Parallel SubBytes and MixColumns over 6 to137 cores. They tested 16 different versions of online and offline key expansion AES cipher. The first small grain implementation for AES was implemented with six to one hundred and seven cores for offline key expansion, as well as eight to one hundred and thirty-seven cores for online key expansion. The authors studied none pipelined silicon-based AES accelerator which offers a throughput of 2.29 Gbps, and the pipelined version offers 8 Gbps throughput. They also studied many AES accelerators; their throughputs range from 2 to 73.7 Gbps. They noticed that the large grain implementation best fits with 8 to 137 cores. The throughput per unit of chip area becomes 3.5-15.6 times higher and the energy efficiency becomes 8.1-18.2 times better. In addition, the throughput of the design was 2.0 times higher compared with the TIDSPC6201. In terms of energy efficiency, the design was 2.9 times higher than the GeForce8800GT [49].

In [50], the authors noticed that AES cryptosystem can be parallelized in many different ways, they have developed a new model for AES that works with larger block and same key size, 200-bits, on the basis of 5 rows and 5 columns and 10 rounds. The authors only modify the mix column transformation component. The results of proposed AES were compared with different implementation of conventional AES of 128, 192, and 256 bits. The comparison was in term of encryption/decryption time and throughput. For 200-bit AES block sizes, the encryption time reduced by 20%, and the time for decryption was increased by 25% compared with conventional AES of key size 128 bits. Regarding the throughput, it was 15%, 20%, and 30% better in their proposed AES-200 compared to AES-128, AES-192, and AES-256 respectively. The authors claim that the security of the proposed 200-bits-AES is better.

Parallel multiprocessor implementation and pipelined design of AES was conducted by [51]. The authors tackled many sources of parallelization in AES algorithm, they pipeline the AES rounds, as well as parallelize MixColumns, and AddRoundKey components. In their parallel AES proposal, 11 stages were used to include the first and last AES rounds, their proposal increased AES throughput. The authors drew the data dependency graph for the AES implementation in order to better understand the exploitation of parallelism with AES. The authors' model provided a 95% improvement. Extending parallelization to Inv_MixColumns transformation improved the performance by 98%.

Given the need for accelerating data encryption and decryption, especially in cloud computing, which by its nature deals with many numbers of users, the authors in [52] implemented Six types of parallel AES algorithms, which are: Coalescent and Sliced GPU, Coalescent & Unsliced

GPU, Uncoalescent GPU, Coalescent & Sliced CPU, Coalescent & Unsliced CPU and Uncoalescent CPU. The authors devoted three algorithms to the GPU, and three to the CPU. For Big data, AES supports many modes: Electronic Code Book, Cipher Feedback, Output Feedback, Counter and Cipher Block Chaining. The authors noted that the counter mode provides more security and parallelism, so they built their six applications in this mode. They evaluated the six parallel applications within either GPU or CPU. Their main motive was to find a cost effective and efficient way to protect electronic commerce on the top of cloud servers. The authors proposed to collect a huge amount of data that comes from many users, split them into segments of the same size, and then encrypt each segment using their six AES algorithms over many cores. This improves the performance of the protection mechanism. Experimental works show the cloud can gain superior performance when using GCS algorithm.

Authors in [53] divided the AES model into three parts: encryption, decryption and key expansion. They proposed a model for paralleling AES algorithm in order to gain efficient data protection on the top of multicore architecture. In their research paper, the authors detailed several points in the AES model, particularly the CBC mode, because it contains many places that can be parallelized. The authors identified parallelized AES parts, and suggested splitting data to be encrypted or decrypted into several fixed-length blocks, and then manipulate them in parallel within multiple core processors. The authors used Verilog language and Xilinx to design Parallel AES system. The simulation results showed that the parallel AES is faster and has more performance compared with serial AES implementations.

4.3 SUMMARY

As we mentioned earlier the enhancements of AES suggestions were tackled both in hardware and software sides. We briefly went through these issues in subsections 4.1 and 4.2. We summarized both hardware and software enhancements in Table 1 and Table 2.

Many of the previous solutions need special equipment, such as FPGA, VHDL or GPU, to increase the AES performance. Some researchers suggested a new layout for AES algorithm in order to reduce AES's execution time, but they didn't check its final security. In this work, a pipelined AES software implementation is suggested, this implementation does not change the overall AES layout, and it does not need any special HW equipment. The solution is developed with the Java programming language, and it is targeted to multicore processors.

Table 1: AES Hardware Implementations.

Author	Method	Throughput Enhancement	Enc/Dec Speed
[42]	HW pipelined AES via FPGAs/Xilinx Virtex1000.	18.5 Gbits/sec with only 542 slices & 10 RAM blocks.	18.5 Gbits/sec.
[45]	AES design & implementation based on Xilinx Virtex XCV600 over FPGA and VHDL.	Throughput reaches 352 Mbit/Sec	352 Mbit/Sec.
[46]	128-AES HW over FPGA & VHDL	6% space enhancement Using Xilinx 14.2	NA
[22]	HW design Using FPGA &VHDL	1609Mbit/Sec on XC6vlx240t of Xilinx Virtex Family	1609 Mbit/sec
[43]	pipelined and unrolled 128-AES & use a memory with multiple ports	104.06 Gbps at frequency 813 MHz	Max efficiency is 30.74 Mbps

Table 2: AES Software Implementations.

Author	Method	Throughput Enhancement	Enc/Dec Speed
[47]	Parallel AES via CUDA platform and NVIDIA 8 GPU	14 times compared with previous work	Increased
[48]	Parallel AES on GPU via CUDA	Increased	7x speedup
[49]	16 implementations and online/offline key expansion	range from 2 to 73.7 Gbps.	3.5-15.6 times higher
[50]	200 bit block size and modifying mix column function	15%, 20%,and 30% better in AES-200.	enc time is reduced by 20% and dec time is increased by 25% .
[51]	pipeline rounds, parallelize MixCol & AddRoundKey components	95% to 98% improvement	NA
[52]	Six parallel AES algorithms using either GPU or CPU parallelism	Improved	GCS has the best performance
[53]	Parallel many parts of AES on multicore processes	Improved	Increased

5.0 THE PROPOSED PIPELINED IMPLEMENTATION OF THE AES

In this section, we present the proposed software pipelined implementation of the AES in order to achieve better throughput using minimum number of threads. The original AES consists of four main operations: AddRoundKey (ARK), SubBytes (SB), ShiftRows (SR) and MixColumns (MC). These operations are applied repeatedly to 128 bits of input message blocks. Figure 5 shows the Pseudo code for the AES cipher, which summarizes the AES whole process. As been noticed from Figure 6 and Figure 7, the four main operations are connected together in a form of chain, where these operations should run in a specific sequence. Each operation sends its processed intermediate cipher block as input to the operation that follows it in the chain. The output of the last operation is returned and sent back to the first operation in the chain. These events are repeated as many times as needed by the AES algorithm.

From parallel computing perspectives, a pipeline is a set of processing operations connected together like a chain, where the output of one operation is the input of the next one in the chain. In pipelined computing, these operations are overlapped and executed in parallel. In our implementation, we assigned the four operations into four threads. The four threads are supposed to work together in parallel on a multicore system when it has a task to do.

To increase the performance of AES implementation, we use the pipelining concept, whereby multiple blocks are processed and overlapped in execution. We created four threads, named: AddRoundKey Thread (ARKT), SubBytes Thread (SBT), ShiftRows Thread (SRT) and MixColumns Thread (MCT). These names reflect the operations that each thread supposed to perform. Each thread connected with a Synchronous Queue named SQAddRoundKey (SARK), SQSubBytes (SSB), SQShiftRows (SSR) and SQMixColumns (SMC). These queues represent the intermediate connections between threads. The reason behind using Synchronous Queues (SQs) is to emphasize that the process of inserting element to the queue by a thread must be followed by the process of removing that element by another thread and vice versa [java doc]. Figure 8 depicts the interaction between threads via SQs.

The implementation of Pipelined AES divides the message that needs to be encrypted into blocks of specific length, ex 128 bits. We name these blocks B1, B2, B3 ...etc. Then these blocks are passed on the four threads one after the other as programmed in the sequential implementation. The difference here is that we deal with threads rather than functions. Suppose that each thread receives a block, it processes the block, it produces a new value for the block and pass it to the next thread in the chain.

```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[0, Nb-1])

   for round = 1 step 1 to Nr-1
      SubBytes(state)
      ShiftRows(state)
      MixColumns(state)
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
   end for

   SubBytes(state)
   ShiftRows(state)
   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

   out = state
end
```

Figure 9: Pseudo Code for AES Cipher [20][54].

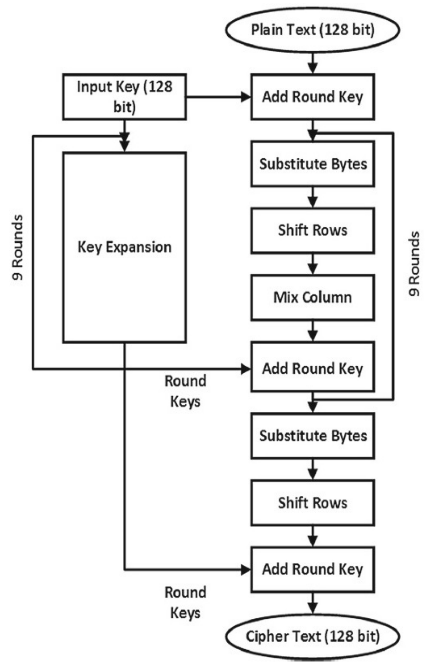


Figure 10: AES Flow Chart [43].

The process starts when ARKT receives B1, process it and then insert the result of B1 into SARK queue, this triggers SBT to take inserted B1 and process it, at the same time, ARKT receives

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

new block B2. After that, SBT inserts the new value of B1 into SSB queue, which in turn triggers the SRT to start working on the B1. Meanwhile SBT is working with B2, ARKT has started working with new block B3. Figure 11 shows the timeline for the four working threads, note that B1 represent block number one and the it1 represents iteration number one, and so on.

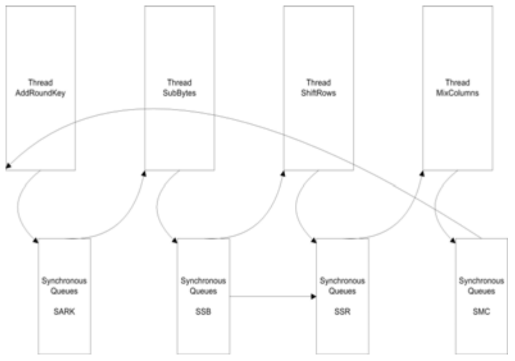
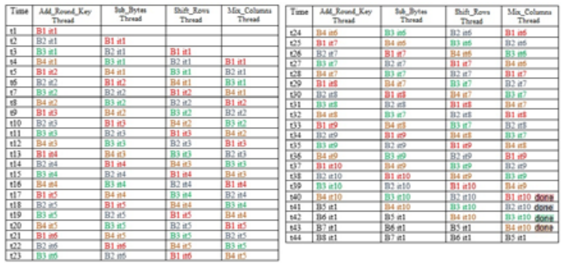


Figure 12: Interaction between Threads Via SQs.



Time	Add_Round_Key Thread	Sub_Bytes Thread	Shift_Rows Thread	Mix_Columns Thread
t1	B1 it1			
t2	B2 it1	B1 it1		
t3	B3 it1	B2 it1	B1 it1	
t4	B4 it1	B3 it1	B2 it1	B1 it1
t5	B1 it2	B4 it1	B3 it1	B2 it1
t6	B2 it2	B1 it2	B4 it1	B3 it1
t7	B3 it2	B2 it2	B1 it2	B4 it1
t8	B4 it2	B3 it2	B2 it2	B1 it2
t9	B1 it3	B4 it2	B3 it2	B2 it2
t10	B2 it3	B1 it3	B4 it2	B3 it2
t11	B3 it3	B2 it3	B1 it3	B4 it2
t12	B4 it3	B3 it3	B2 it3	B1 it3
t13	B1 it4	B4 it3	B3 it3	B2 it3
t14	B2 it4	B1 it4	B4 it3	B3 it3
t15	B3 it4	B2 it4	B1 it4	B4 it3
t16	B4 it4	B3 it4	B2 it4	B1 it4
t17	B1 it5	B4 it4	B3 it4	B2 it4
t18	B2 it5	B1 it5	B4 it4	B3 it4
t19	B3 it5	B2 it5	B1 it5	B4 it4
t20	B4 it5	B3 it5	B2 it5	B1 it5
t21	B1 it6	B4 it5	B3 it5	B2 it5
t22	B2 it6	B1 it6	B4 it5	B3 it5
t23	B3 it6	B2 it6	B1 it6	B4 it5
t24	B4 it6	B3 it6	B2 it6	B1 it6
t25	B1 it7	B4 it6	B3 it6	B2 it6
t26	B2 it7	B1 it7	B4 it6	B3 it6
t27	B3 it7	B2 it7	B1 it7	B4 it6
t28	B4 it7	B3 it7	B2 it7	B1 it7
t29	B1 it8	B4 it7	B3 it7	B2 it7
t30	B2 it8	B1 it8	B4 it7	B3 it7
t31	B3 it8	B2 it8	B1 it8	B4 it7
t32	B4 it8	B3 it7	B2 it8	B1 it8
t33	B1 it9	B4 it8	B3 it7	B2 it8
t34	B2 it9	B1 it9	B4 it8	B3 it7
t35	B3 it9	B2 it9	B1 it9	B4 it8
t36	B4 it9	B3 it9	B2 it9	B1 it9
t37	B1 it10	B4 it9	B3 it9	B2 it9
t38	B2 it10	B1 it10	B4 it9	B3 it9
t39	B3 it10	B2 it10	B1 it10	B4 it9
t40	B4 it10	B3 it10	B2 it10	B1 it10 done
t41	B5 it1	B4 it10	B3 it10	B2 it10 done
t42	B6 it1	B5 it1	B4 it10	B3 it10 done
t43	B7 it1	B6 it1	B5 it1	B4 it10 done
t44	B8 it1	B7 it1	B6 it1	B5 it1

Figure 13: Time Slices of the Threads and their Tasks.

As seen in Figure 14, at time t1: ARKT receives the first block B1, process it, and send the result to SBT at time t2, at the same time, ARKT receives block B2. At time t3, the output from SBT goes to SRT, SBT receives block B2 from ARKT, meanwhile, ARKT receives block B3. At time t4, MCT is busy working with block B1, SRT is working with block B2, SBT is working with block B3 and ARKT is working with block B4. After that, MCT sends back block B1 to ARKT, and these actions are repeated for many iterations (Nr). To clarify this, at the end of time t5, MCT sends block B1 to ARKT in order to start processing it for the second iteration at time t5, during this time, block B1 is processed five times. After (Nr *4) times, block B1 is ready. During the next time unit, block B2 is ready, then block B3 is ready and after that, block B4 becomes ready. Each [n*(Nr*4)] iterations, nth four cipher blocks are completed, where n equals the number of blocks.

If the four threads are working simultaneously, and if each thread takes the same amount of time to finish its work, then the speedup of the pipelined AES compared with sequential AES should equal to 4, assuming that each thread is working on a separated core.

It should be noted that each thread needs (Nr+1) keys in the process of encryption. These keys should be produced before the whole encryption process starts. These keys need to be distributed to all threads. This process is called key expansion and scheduling. Figure 15 shows the pseudo code for this process.

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
   word  temp

   i = 0

   while (i < Nk)
      w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
      i = i+1
   end while

   i = Nk

   while (i < Nb * (Nr+1)]
      temp = w[i-1]
      if (i mod Nk = 0)
         temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
      else if (Nk > 6 and i mod Nk = 4)
         temp = SubWord(temp)
      end if
      w[i] = w[i-Nk] xor temp
      i = i + 1
   end while
end
```

Figure 16: key expansion and scheduling process [55].

6.0 PIPELINED AES IMPLEMENTATION ANALYSIS

Pipelining block processing is one way to improve the AES performance, it is achieved by interleaving the encryption process of multiple blocks within multiple thread of control simultaneously. The four fundamental operations (ARK, SB, SR and MC) are assigned into four threads. With multicore system, the simultaneous run of the four threads can be achieved, if each thread has a block to work with. The throughput of the sequential AES encryption algorithm is measured by computing number of plain blocks being completed within a certain amount of time. The latency time in converting each plain text into cipher text is measured by the time needed to process each block individually multiplied by the number of blocks, since the individual time is the same for all blocks.

Each step in the pipeline process is called a pipeline stage, and a stage time is the time required to complete a single pipeline stage. We have four threads, each do a certain task, then each is considered a stage, so we have four stages in our context of this paper. These stages named: AddRoundKey stage (SARK), SubBytes stage (SSB), ShiftRows stage (SSR) and MixColumns stage (SMC). We call the time required for AddRoundKey stage tARK, the time required for SubBytes stage tSB, the time required for ShiftRows stage tSR, and the time required for MixColumns stage tMC. Equation 1 represents the time for serial computation of the AES algorithm:

$$ts_{AES} = t_{RB} + (t_{ARK} + T_{SB} + t_{SR} + t_{MC}) * Nr * NB \quad (1)$$

Where ts_{AES} is the serial execution time of the AES, t_{RB} is the time needed to read plain text and divide it into blocks, Nr is the number of iterations and NB is the number of blocks.

Each block needs a processing time equal to the sum of elapsed time for four major operations multiplied by the number of iterations Nr. The AES pipelined version proposed in this paper has bigger latency to convert each individual block of plain text into a block of cipher text, because there is a dependency between threads, leading to a time equal to the largest values among tARK, tSB, tSR and tMC. The following formula depicts this fact.

$$tp_{AES} = t_{RB} + (\mathrm{Ps} - 1 + (\mathrm{NB} * \mathrm{Nr})) * max(t_{ARK}, T_{SB}, t_{SR}, t_{MC}) \quad (2)$$

Where tp_{AES} is the Pipelined AES time, Ps is the number of pipeline stages, which equals to 4 in this paper, Nr is the number of iterations to process all blocks and NB is the number of blocks. Max (t_{ARK}, t_{SB}, t_{SR}, t_{MC}) is the maximum time of these stages.

The implementation of pipelined AES overlaps the execution of different blocks simultaneously. Our Pipelined AES works with multiple threads equal to four in sequence, so the result of the first thread is an input to the second thread, the output of the second thread is fed to the third thread, the output of the third thread is fed to the fourth thread, and the output of the fourth thread is fed back to the first thread, these four operations is iterated Nr times.

As we know from pipeline implementations, the speed of the stages becomes the same as the slowest stage, which is equal to the maximum time of t_{ARK}, t_{SB}, t_{SR} and t_{MC}. In other words, the four basic operations, SubBytes, AddRoundKey, ShiftRows and MixColumns required for the encryption process have a specific time. These operations are employed for four threads, so each thread has a specific time to finish its work on each block. If these times are not equal and the threads are working at the same time, the times will unify for the greatest time in them. This time is called time pipeline's slot (tps). The time needed to encrypt a block individually becomes equal to the value of tps multiplied by the number of iterations Nr multiplied by Ps [56], as equation 2 depicts. The number of time slots required to end the encryption process for a number of blocks is expressed in equation 3.

$$\text{Number of tps} = \mathrm{Ps} - 1 + \mathrm{NB} * \mathrm{Nr} \quad (3)$$

Equation 3 depicts the fact that after Ps-1 time slots all four threads operate simultaneously with different blocks. These Ps-1 times are needed to fill the pipeline chain. Every Nr * Ps pipeline time slots, four block ciphers are completed. Equation 4 shows the speedup (S) of the proposed pipeline implementation.

$$S = \frac{ts_{AES}}{tp_{AES}} \quad (4)$$

$$S = \frac{t_{RB} + (t_{ARK} + t_{SB} + t_{SR} + t_{MC}) * Nr * NB}{t_{RB} + ((Ps - 1 + Nr * NB) * tps)} \quad (5)$$

Where tps is the maximum time of t_{ARK}, t_{SB}, t_{SR} and t_{MC}. The maximum speedup (MS) would be obtained when the NB approach to infinity, equation 6 reflects the formula of MS.

$$MS = \lim_{NB\to\infty}\left(\frac{t_{RB} + (t_{ARK} + t_{SB} + t_{SR} + t_{MC}) * Nr * NB}{t_{RB} + ((Ps - 1 + Nr * NB) * tps)}\right) (6)$$

We ignore t_{RB}, because this serial part considered constant value or lower degree term with respect to the variable NB, due to it is being read one time. This t_{RB} value is shared between serial code and the pipelined version, so we will concentrate only on the portion of the serial code that will be parallelized.

$$MS = \lim_{NB\to\infty}\left(\frac{(t_{ARK} + t_{SB} + t_{SR} + t_{MC}) * Nr * NB}{(Ps - 1 + Nr * NB) * tps}\right) (7)$$

We ignore Ps-1 because this value is too small with respect to Nr * NB * tps.

$$MS = \lim_{NB\to\infty}\left(\frac{(t_{ARK}+t_{SB}+t_{SR}+t_{MC})*Nr*NB}{Nr*NB*tps}\right)(8)$$

From equation 8, we can remove Nr *NB from nominator and denominator. The result is seen in equation 9.

$$MS = \left(\frac{t_{ARK}+t_{SB}+t_{SR}+t_{MC}}{tps}\right)(9)$$

By substituting the value of tps, the final equation becomes

$$MS = \left(\frac{t_{ARK}+t_{SB}+t_{SR}+t_{MC}}{max(t_{ARK},t_{SB},t_{SR},t_{MC})}\right)(10)$$

We notice from equation 10 that the maximum speedup depends on the slowest time of the four operations t_{ARK}, t_{SB}, t_{SR} and t_{MC}. So if these times are equal, then the maximum speedup is equal to 4.

7.0 EXPERIMENT SETUP

The specifications in Table 3 were used to test our pipeline implementation. We use an Intel based Dell Inspiron 15 5000 series Intel® core™ i7-6500U machine with CPU Speed of 2.5 GHz and 16 GB RAM. Table 4 shows the time execution in nanosecond according to the Intel based Dell Inspiron machine.

Table 3: HW specifications Used to Test the Pipeline Implementation

Machine Type		CPU speed	RAM
Dell Inspiron 15 5000 series Intel ® core™ i7-6500U		2.5GHz	16 GB
Time of (t_{ARK}, t_{SB}, t_{SR}, t_{MC}) in nanoseconds			
t_{ARK}	t_{SB}	t_{SR}	t_{MC}
790 ns	1185 ns	1975 ns	5536 ns

From these readings, we can expect that the maximum speedup is equal to 1.7. See the following substitutions from equation 10.

$$MS = \left(\frac{790\ +1185\ +1975\ +5536}{max(790,1185,1975,5536)}\right) = 1.7$$

We run both the serial version and our pipeline version on different file sizes, we obtained the following results as been depicted in Table 3.

As shown in Table 4, as the number of blocks increase the improvement degree increases and reached to maximum theoretical measures. The chart in Figure 17 shows the enhancement in the performance with respect to the number of blocks.

Table 4: The Run of Serial and Pipelined AES Algorithms

File size in kilo Byte	Sequential run in microsecond	Our pipeline run in microsecond	Speedup
20	78000500	62400400	1.25
40	109200700	78000500	1.4
80	191601100	156001000	1.2
160	234001500	187201200	1.25
5639	639604100	390002500	1.64

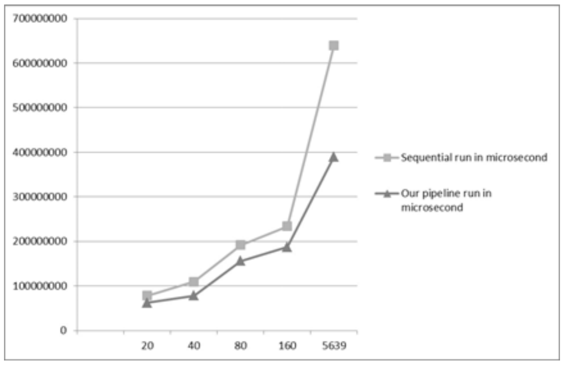


Figure 17: Enhances in The Performance.

Compared to the achievements made to improve the speed of AES by many researchers, what distinguishes our work is that it does not need expensive equipment or software such as GPU, FPGA or CUDA.

8.0 Conclusion

The Advanced Encryption Standard (AES) is a symmetric block for encryption and decryption algorithm, it mainly composed of four functions AddRoundKey, SubBytes, ShiftRows and MixColumns, these functions must be implemented in a specific order and for Nr times for each block. Four threads have been employed so that each of them is dedicated to carrying out one of the four main AES functions. They are prepared to work in parallel. We analyzed pipelined implementation and showed that the result has a significant increase in the performance. Our technique enhances the performance of the algorithm by 1.7 times compared with serial one. The contribution of this pipelined AES implementation is that the proposed implementation helps the majority of users in the world to accelerate the work of AES without the need for special equipment such as GPU, FPGA or CUDA.

REFRENCES:

[1] W. Madsen, "Trust in Cyberspace," *Netw. Secur.*, vol. 1999, no. 11, pp. 18–19, 1999.

[2] H. Nissenbaum, "Where computer security meets national security," *Ethics Inf. Technol.*, vol. 7, no. 2, pp. 61–73, 2005.

[3] J. Bielby *et al.*, "Protecting Internet Traffic: Security Challenges and Solutions," 2017.

[4] A. Nadeem and M. Y. Javed, "A Performance Comparison of Data Encryption Algorithms," *2005 Int. Conf. Inf. Commun. Technol.*, no. September, pp. 84–89, 2005.

[5] M. Barhoush and J. W. Atwood, "Requirements for enforcing digital rights management in multicast content distribution," *Telecommun. Syst.*, vol. 45, no. 1, pp. 3–20, 2010.

[6] W. Stallings, *Cryptography and Network Security, 4/E*. Pearson Education India, 2006.

[7] G. J. Popek and C. S. Kline, "Encryption and secure computer networks," *ACM Comput. Surv.*, vol. 11, no. 4, pp. 331–356, 1979.

[8] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, "Performance comparison of the AES submissions." 1999.

[9] T. Karygiannis and L. Owens, "Wireless network security," *NIST Spec. Publ.*, vol. 800, p. 48, 2002.

[10] M. Hendry, *Smart card security and applications*. Artech House, 2001.

[11] A. Odeh, S. R.Masadeh, and A. Azzazi, "A performance evaluation of common encryption techniques with secure watermark system (sws)," *Int. J. Netw. Secur. Its Appl.*, vol. 7, no. 3, pp. 31–38, 2015.

[12] N. H. Zakaria, R. Mahmod, N. I. Udzir, and Z. A. ZUKARNAIN, "ENHANCING ADVANCED ENCRYPTION STANDARD (AES) S-BOX GENERATION USING AFFINE TRANSFORMATION.," *J. Theor. Appl. Inf. Technol.*, vol. 72, no. 1, 2015.

[13] P. Arul and A. Shanmugam, "GENERATE A KEY FOR AES USING BIOMETRIC FOR VOIP NETWORK SECURITY.," *J. Theor. Appl. Inf. Technol.*, vol. 5, no. 2, 2009.

[14] P. Garg, "GENETIC ALGORITHMS, TABU SEARCH AND SIMULATED ANNEALING: A COMPARISON BETWEEN THREE APPROACHES FOR THE CRYPTANALYSIS OF TRANSPOSITION CIPHER.," *J. Theor. Appl. Inf. Technol.*, vol. 5, no. 4, 2009.

[15] D. P. Leech, M. W. Chinworth, G. G. Payne, and C. M. Waychoff, "The Economic Impacts of NIST's Data Encryption Standard (DES) Program," 2001.

[16] V. Patel, R. C. Joshi, and A. K. Saxena, "FPGA IMPLEMENTATION OF DES USING PIPELINING CONCEPT WITH SKEW CORE KEY-SCHEDULING.," *J. Theor. Appl. Inf. Technol.*, vol. 5, no. 3, 2009.

[17] G. Singh, "A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security," *Int. J. Comput. Appl.*, vol. 67, no. 19, pp. 975–8887, 2013.

[18] R. Bhanot and R. Hans, "A review and comparative analysis of various encryption algorithms," *Int. J. Secur. Its Appl.*, vol. 9, no. 4, pp. 289–306, 2015.

[19] J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*. 2002.

[20] V. Rijmen and J. Daemen, "Advanced encryption standard," *Proc. Fed. Inf. Process. Stand. Publ. Natl. Inst. Stand. Technol.*, pp. 19–22, 2001.

[21] "Advanced Encryption standard," 2002.

[22] K. P. Singh and S. Dod, "An Efficient Hardware design and Implementation of Advanced Encryption Standard (AES) Algorithm," no. February, pp. 5–9, 2016.

[23] M. M. Thulasimani, "Design And Implementation of Reconfigurable Rijndael Encryption Algorithms For Reconfigurable Mobile Terminals," *Int. J. Comput. Sci. Eng.*, vol. 02, no. 04, pp. 1003–1011, 2010.

[24] N. A. Kofahi, "An empirical study to compare the performance of some symmetric and asymmetric ciphers," *Int. J. Secur. its Appl.*, vol. 7, no. 5, pp. 1–16, 2013.

[25] S. Agarwal, "Image encryption techniques using fractal function: a review," *Int. J. Comput. Sci. Inf. Technol.*, vol. 9, no. 2, pp. 53–68, 2017.

[26] D. Pandya and K. R. Narayan, "Brief History of Encryption," vol. 131, no. 9, pp. 28–31.

[27] M. Nagendra and M. Chandra Sekhar, "Performance improvement of advanced encryption algorithm using parallel computation," *Int. J. Softw. Eng. its Appl.*, vol. 8, no. 2, pp. 287–296, 2014.

[28] N. A. Rathi and S. R. Gupta, "Analysis of Security mechanism in E-commerce transaction," vol. 5, no. 1, pp. 131–135, 2016.

[29] A. Barnes, R. Fernando, K. Mettananda, and R. Ragel, "Improving the throughput of the AES algorithm with multicore processors," *2012 IEEE 7th Int. Conf. Ind. Inf. Syst. ICIIS 2012*, no. August, 2012.

[30] M. E. Smid and D. K. Branstad, "Data encryption standard: past and future," *Proc. IEEE*, vol. 76, no. 5, pp. 550–559, 1988.

[31] D. Coppersmith, "The Data Encryption Standard (DES) and its strength against attacks," *IBM J. Res. Dev.*, vol. 38, no. 3, pp. 243–250, 1994.

[32] S. Bhati, A. Bhati, and S. K. Sharma, "A New Approach towards Encryption Schemes : Byte – Rotation Encryption Algorithm," vol. II, no. 009352118885, pp. 24–27, 2012.

[33] P. K. Dey and T. K. Dey, "ANALYSIS OF THE SECURITY OF AES, DES, 3DES AND IDEA NXT ALGORITHM," *Int. J. Eng. Sci. Res. Technol.*

[34] J. Foti, "Status of the advanced encryption standard (AES) development effort," in *Proc. 21st NIST-NCSC National Information Systems Security Conference*, 1998, pp. 549–554.

[35] E. Biham and A. Shamir, *Differential cryptanalysis of the data encryption standard*. Springer Science & Business Media, 2012.

[36] S. Soni, H. Agrawal, and M. Sharma, "Analysis and Comparison between AES and DES Cryptographic Algorithm," *Ijeit*, vol. 2, no. 6, pp. 362–365, 2012.

[37] L. Thulasimani and M. Madheswaran, "A single chip design and implementation of aes-128/192/256 encryption algorithms," *Int. J. Eng. Sci. Technol.*, vol. 2, no. 5, pp. 1052–1059, 2010.

[38] A. Srivastava and N. R. Venkataraman, "AES-128 Performance in Tinyos with CBC Algorithm," vol. 7, no. 5, pp. 40–49, 2013.

[39] A. Berent, "Aes (advanced encryption standard) simplified," 2003.

[40] J. Daemen and V. Rijmen, "AES proposal: Rijndael," 1999.

[41] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[42] F. Standaert, G. Rouvroy, J. Quisquater, and J. Legat, "Efficient Implementation of Rijndael Encryption in Reconfigurable Hardware : Improvements and Design Tradeoffs," *Cryptogr. Hardw. Embed. Syst. - CHES 2003*, vol. 2779, pp. 334–350, 2003.

[43] M. B. Chellam and R. Natarajan, "AES hardware accelerator on FPGA with improved throughput and resource efficiency," *Arab. J. Sci. Eng.*, pp. 1–18, 2017.

[44] M. Shakir, A. B. Abubakar, Y. Bin Yousoff, and M. Sheker, "IMPROVEMENT KEYS OF ADVANCED ENCRYPTION STANDARD (AES) RIJNDAEL_M.," *J. Theor. Appl. Inf. Technol.*, vol. 86, no. 2, 2016.

[45] P. B. Ghewari and A. B. Chougule, "Efficient Hardware Design and Implementation of AES Cryptosystem," *Int. J. Eng. Sci. Technol.*, vol. 2, no. 3, pp. 213–219, 2010.

[46] L. Anjali, "An Efficient Hardware FPGA Implementation of AES-128 Cryptosystem Using Vedic Multiplier and Non LFSR," vol. 3, no. 5, pp. 842–846, 2014.

[47] A. Di Biagio, A. Barenghi, G. Agosta, and G. Pelosi, "Design of a parallel AES for graphics hardware using the CUDA framework," *IPDPS 2009 - Proc. 2009 IEEE Int. Parallel Distrib. Process. Symp.*, 2009.

[48] D. Le, J. Chang, X. Gou, A. Zhang, and C. Lu, "Parallel AES algorithm for fast data encryption on GPU," *ICCET 2010 - 2010 Int. Conf. Comput. Eng. Technol. Proc.*, vol. 6, pp. 1–6, 2010.

[49] B. Liu and B. M. Baas, "Parallel aes encryption engines for many-core processor arrays," *IEEE Trans. Comput.*, vol. 62, no. 3, pp. 536–547, 2013.

[50] R. Pahal and V. Kumar, "Efficient Implementation of AES," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 3, no. 7, pp. 290–293.

[51] G. F. Elkabbany, H. K. Aslan, and M. N. Rasslan, “A design of a fast parallel-pipelined implementation of AES: advanced encryption standard,” *arXiv Prepr. arXiv1501.01427*, 2015.

[52] X. Fei, K. Li, W. Yang, and K. Li, “Practical parallel {AES} algorithms on cloud for massive users and their performance evaluation,” *Concurr. Comput. Pract. Exp.*, vol. 28, no. 16, pp. 4246–4263, 2016.

[53] R. Patel and S. Kanjariya, “Design of Parallel Advanced Encryption Standard (AES) Algorithm,” vol. 4, no. 3, pp. 219–222, 2015.

[54] D. Selent, “Advanced encryption standard,” *Rivier Acad. J.*, vol. 6, no. 2, pp. 1–14, 2010.

[55] S. Heron, “Advanced encryption standard (AES),” *Netw. Secur.*, vol. 2009, no. 12, pp. 8–12, 2009.

[56] J. L. Hennessy and D. A. Patterson, *Computer architecture: a quantitative approach*. Elsevier, 2011.