RESEARCH ARTICLE OPEN ACCESS

# Encryption Algorithms - Parallelization

Shweta Kumari[1], Abhishek Kumar[2]
School of Computing Science and Engineering
Galgotias University
Greater Noida
U.P-India

**ABSTRACT**
Most cryptographic algorithms function more efficiently when implemented in hardware than in software running on single processor. However, systems that use hardware implementations have significant drawbacks: they are unable to respond to flaws discovered in the implemented algorithm or to changes in standards. As an alternative, it is possible to implement cryptographic algorithms in software running on multiple processors. In this paper, encryption has been implemented with parallel processor and the result has been compared.
***Keywords:-*** Cryptography, Parallelization, Comparison.

## I. INTRODUCTION

The emergence of the Internet as a trusted medium for commerce and communication has made cryptography an essential component of modern information systems. Cryptography provides the mechanisms necessary to implement accountability, accuracy, and confidentiality in communications [1]. As demands for secure communication bandwidth grow, efficient cryptographic processing will become increasingly vital to good system performance. To introduce cryptography, an understanding of issues related to information security in general is necessary. Information security manifests itself in many ways according to the situation and requirement. Regardless of who is involved, to one degree or another, all parties to a transaction must have confidence that certain objectives associated with information security have been met. Over the centuries, an elaborate set of protocols and mechanisms has been created to deal with information security issues when the information is conveyed by physical documents. Often the objectives of information security cannot solely be achieved through mathematical algorithms and protocols alone, but require procedural techniques and abidance of laws to achieve the desired result. For example, privacy of letters is provided by sealed envelopes delivered by an accepted mail service. The physical security of the envelope is, for practical necessity, limited and so laws are enacted which make it a criminal offense to open mail for which one is not authorized. It is sometimes the case that security is achieved not through the information itself but through the physical document recording it. For example, paper currency requires special inks and materials to prevent counterfeiting. Achieving information security in an electronic society requires a vast array of technical and legal skills. There is, however, no guarantee that all of the information security objectives deemed necessary can be adequately met. The technical means is provided through cryptography.

## II. CRYPTOGRAPHY GOALS

Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication [1]. Cryptography is not the only means of providing information security, but rather one set of techniques.

The following four cryptographic goals form a framework from which other goals are derived:

### *A. Confidentiality*

It is a service used to keep the content of information from all but those authorized to have it.

### *B. Data integrity*

It is a service which addresses the unauthorized alteration of data. 3.

### *C. Authentication*

It is a service related to identification.

### *C. Non-repudiation*

It is a service which prevents an entity from denying previous commitments or actions. When disputes arise due to an entity denying that certain
actions were taken, a means to resolve the situation is necessary.

## III. SYMMETRIC-KEY CRYPTOGRAPHY

Symmetric-key cryptography, also called secret key cryptography, is the most intuitive kind of cryptography. It involves the use of a secret key known only to the participants of the secure communication. Symmetric-key cryptography can be used to transmit information over an insecure channel, but it has also other uses, such as secure storage on insecure media or strong mutual authentication. In symmetric-key

cryptography [2], the key must be shared by both the sender and the receiver. The sender applies the encryption function using the key to the plaintext to produce the cipher text. The cipher text is sent to the receiver, who then applies the decryption function using the same shared key. Since the plaintext cannot be derived from the cipher text without knowledge of the key, the cipher text can be sent over public networks such as the Internet. Therefore, symmetric key cryptography is characterized by the use of a single key to perform both the encrypting and decrypting of data. Since the algorithms are public knowledge, security is determined by the level of protection afforded the key. If key is kept secret, both the secrecy and authentication services are provided. Secrecy is provided, because if the message is intercepted, the intruder cannot transform the cipher text into its plaintext format.
Assuming that only two users know the key, authentication is provided because only a user with the key can generate cipher text that a recipient can transform into meaningful plaintext.

The main aims of this thesis are to implement encryption in serial, find out its performance on single processor and compare the performance with parallel implementations of encryption with 2, 4, and 8 processors varying different parameters such as key size, number of rounds and extended key size, and show how parallel implementation of the AES offers better performance yet flexible enough for cryptographic algorithms.

## IV CONVENTIONAL ENCRYPTION

**Cryptography** (from Greek *kryptÃ³s*, "hidden", and *grÃ¡phein*, "to write") is generally understood to be the study of the principles and techniques by which information is converted into an encrypted version that is difficult (ideally impossible) for any unauthorized person to convert to the original information, while still allowing the intended reader to do so. In fact, cryptography covers rather more than merely encryption and decryption. It is, in practice, a specialized branch of information theory with substantial additions from other branches of mathematics. Cryptography is probably the most important aspect of communications security and is becoming increasingly important as a basic building block for computer security [3].
There are, in general, two types of cryptographic schemes typically used to accomplish these goals:

***A. Secret Key*** (or **symmetric** or **conventional**) cryptography and
***B. Public Key*** (or asymmetric) cryptography.

In **symmetric-key** cryptography, an algorithm is used to scramble the message using a secret key in such a way that it becomes unusable to all except the ones that have access to that secret key. The most widely known symmetric cryptographic algorithm is DES, developed by IBM in the seventies. It uses a key of 56 bits and operates on chunks of 64 bits at a time. In **public key** cryptography [4], algorithms use two different keys: a **private** and a **public** one. A message encrypted with a private key can be decrypted with its public key (and vice versa). The two basic building blocks of all encryption techniques are **substitution** and **transposition**.

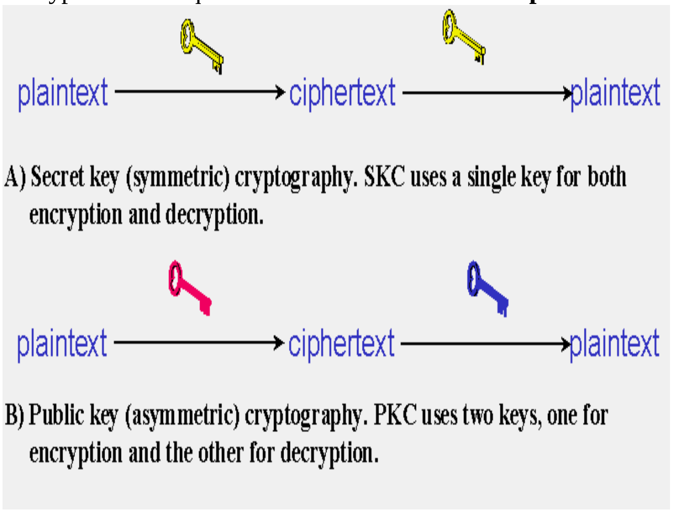
plaintext
ciphertext
plaintext
A) Secret key (symmetric) cryptography. SKC uses a single key for both encryption and decryption.
plaintext
ciphertext
plaintext
B) Public key (asymmetric) cryptography. PKC uses two keys, one for encryption and the other for decryption.

**Figure 1 Two types of cryptography**

Symmetric-key cryptography schemes are generally categorized as being either **stream ciphers** or **block ciphers** [5]. Stream ciphers operate on a single bit (byte or computer word) at a time, and implement some form of feedback mechanism so that the key is constantly changing. A block cipher is so-called because the scheme encrypts one block of data at a time using the same key on each block. In general, the same plaintext block will always encrypt to the same cipher text when using the same key in a block cipher whereas the same plaintext will encrypt to different cipher text in a stream cipher. **Stream ciphers** come in several flavours but two are widely used. **Self-synchronizing stream** *ciphers* calculate each bit in the key stream as a function of the previous *n* bits in the key stream. It is termed "self-synchronizing" because the decryption process can stay synchronized with the encryption process merely by knowing how far into the *n*-bit key stream it is. One problem is error propagation; a garbled bit in transmission will result in *n* garbled bits at the receiving side [6].

**Synchronous stream ciphers** generate the key stream in a fashion independent of the message stream but by using the same key stream generation function at sender and receiver. While stream ciphers do not propagate transmission errors, they are, by their nature, periodic so that the key stream will eventually repeat [7]. **Block ciphers** can operate in one of several modes; the following four are the most important:

***A. Electronic Codebook (ECB)*** *mode* is the simplest most obvious application: the secret key is used to encrypt the plaintext block to form a cipher text block. Two identical plaintext blocks, then, will always generate the same cipher text block. Although this is the most common mode of block ciphers, it is susceptible to a variety of brute-force attacks.

***B. Cipher Block Chaining (CBC)*** *mode* adds a feedback mechanism to the encryption scheme. In CBC, the plaintext is exclusively-ORed (XORed) with the previous ciphertext block prior to encryption. In this mode, two identical blocks of plaintext never encrypt to the same cipher text.

***C. Cipher Feedback (CFB)*** *mode* is a block cipher implementation as a self-synchronizing stream cipher. CFB mode allows data to be encrypted in units smaller than the block size, which might be useful in some applications such as encrypting interactive terminal input. In case of 1-byte CFB mode, for example, each incoming character is placed into a shift register the same size as the block, encrypted, and the block transmitted [8]. At the receiving side, the ciphertext is decrypted and the extra bits in the block (i.e., everything above and beyond the one byte) are discarded.

***D. Output Feedback (OFB)*** *mode* is a block cipher implementation conceptually similar to a synchronous stream cipher. OFB prevents the same plaintext block from generating the same ciphertext block by using an internal feedback mechanism that is independent of both the plaintext and ciphertext bit streams.

## V. MODEL OF SYMMETRIC-KEY CRYPTOSYSTEM

A symmetric or conventional encryption scheme has five ingredients (Figure 2)**:**
***Plaintext /Message:*** This is the original intelligible message or data that is fed into the algorithm as input [9].
***Encryption Algorithm:*** The encryption algorithm performs various substitution and transformation on the plaintext.
***Secret Key:*** The secret key is also the input to the encryption algorithm. The key is a value independent of the plaintext. The algorithm will produce a different output depending on the specific key being used at the time. The exact substitutions and transformations performed by the algorithm depend on the key.
***Ciphertext:*** This is the scrambled message produced as output. It depends on the plaintext and secret key. For a given message, two different keys will produce two different ciphertexts.
***Decryption Algorithm:*** This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

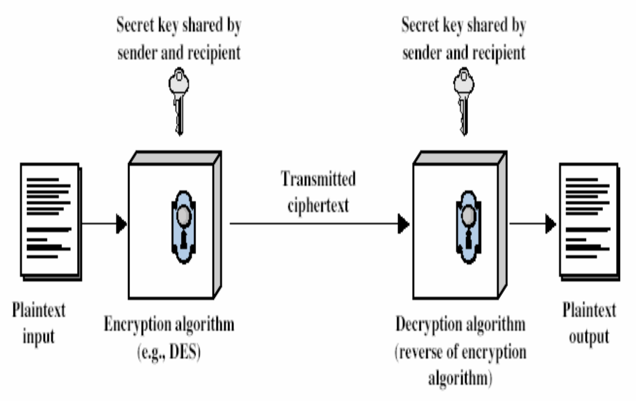
Secret key shared by sender and recipient
Secret key shared by sender and recipient
Transmitted ciphertext
Plaintext input
Encryption algorithm (e.g., DES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

**Figure 2 Model of Symmetric-key Cryptosystem**

There are two requirements for secure use of symmetric-key encryption:
***A.*** A strong encryption algorithm. At a minimum, the algorithm to be such that an opponent who knows the algorithm and has access to one or more ciphertexts would be unable to decipher the ciphertext or figure out the key. This requirement is usually stated in a stronger form: The opponent should be unable to decrypt ciphertext or discover the key even if he or she is in possession of a number of ciphertexts together with the plaintext that produced each ciphertext.
***B.*** Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure. If some can discover the key and knows the algorithm, all communication using this key is readable [10].

### *A. Importance of Symmetric-key Cryptography*

The primary advantage of public-key cryptography is increased security and convenience. Private keys never need to transmitted or revealed to anyone. In a symmetric-key system, by contrast, the symmetric keys must be transmitted (either manually or through a communication channel), and there may be a chance that an enemy can discover the symmetric keys during their transmission. Another major advantage of public-key systems is that they can provide a method for **digital signatures**. Authentication via symmetric-key systems requires the sharing of some symmetric keys and sometimes requires trust of a third party as well. As a result, a sender can repudiate a previously authenticated message by claiming that the shared symmetric key was somehow compromised by one of the parties sharing the symmetric-key. Public-key authentication, on the other hand, prevents this type of repudiation; each user has sole responsibility for protecting his or her private key. This property of public-key authentication is often called **nonrepudiation [11]**.

A **disadvantage** of using public-key cryptography for encryption is **speed**; there are

popular symmetric-key encryption methods that are significantly faster than any currently available public-key encryption method. Nevertheless, public-key cryptography can be used with symmetric-key cryptography to get the best of both worlds. **For encryption**, the best solution is to combine public- and symmetric-key systems in order to get both the security advantages of public-key systems and the speed advantages of symmetric-key systems. The public-key system can be used to encrypt a symmetric- key which is used to encrypt the bulk of a file or message.
Such a protocol is called a *digital envelope*.

### *B. Advantages of symmetric-key cryptography*

1. Symmetric-key ciphers can be designed to have high rates of data throughput.
2. Keys for symmetric-key ciphers are relatively short [12].
3. Symmetric-key ciphers can be employed as primitives to construct various cryptographic mechanisms including pseudorandom number generators, hash functions, and computationally efficient digital signature schemes, to name just a few.
4. Symmetric-key ciphers can be composed to produce stronger ciphers. Simple transformations which are easy to analyze, but on their own weak, can be used to construct strong product ciphers.

### *C. Disadvantages of symmetric-key cryptography*

1. In a two-party communication, the key must remain secret at both ends.
2. In a large network, there are many key pairs to be managed. Consequently, effective key management requires the use of an unconditionally trusted TTP.
3. In a two-party communication between entities A and B, sound cryptographic practice dictates that the key be changed frequently and perhaps for each communication session.

Digital signature mechanisms arising from symmetric-key encryption typically require either large keys for the public verification function or the use of a TTP.

### *D. Advantages of public-key cryptography*

1. Only the private key must be kept secret (authenticity of public keys must, however, be guaranteed).
2. Depending on the mode of usage, a private key/public key pair may remain unchanged for considerable periods of time, e.g., many sessions (even several years).
3. Many public-key schemes yield relatively efficient digital signature mechanisms. The key used to describe the public verification function is typically much smaller than for the symmetric-key counterpart.
4. In a large network, the number of keys necessary may be considerably smaller than in the symmetric-key scenario.

### *E. Disadvantages of public-key encryption*

1. Throughput rates for the most popular public-key encryption methods are several orders of magnitude slower than the best-known symmetric-key schemes.
2. Key sizes are typically much larger than those required for symmetric-key encryption, and the size of public-key signatures is larger than that of tags providing data origin authentication from symmetric-key techniques.

### *F. Summary of comparison*

1. Public-key cryptography facilitates efficient signatures (particularly nonrepudiation)
and key management, and
2. Symmetric-key cryptography is efficient for encryption and some data
integrity applications.

## VI. PARALLEL IMPLEMENTATION OF ENCRYPTION ALGORITHM

The current trend in high performance computing is clustering and distributed computing. In clusters, powerful low cost workstations and/or PCs are linked through fast communication interfaces to achieve high performance parallel computing. Recent increases in communication speeds, microprocessor clock speeds, availability of high performance public domain software including operating system, compiler tools and message passing libraries, make cluster based computing appealing in terms of both high performance computing and cost effectiveness. Parallel computing on clustered systems is a viable and attractive proposition due to the high communication speeds of modern networks [13]. To efficiently use more than one processor in a program, the processors must share data and co-ordinate access to and updating of the shared data. The most popular approach to this problem is to exchange of data through messages between computers.

The MPI (message Passing Interface) approach is considered to be one of the most mature methods currently used in parallel programming mainly due to the relative simplicity of using the method by writing a set of library functions or an API (Application Program Interface) callable from C, C++ or Fortran Programs. MPI was designed for high performance on both massively parallel machines and clusters. Today, MPI is considered a de facto standard for message passing in the parallel-computing paradigm. For implementing the encryption algorithm in parallel, the MPI based cluster is used in the present chapter.

The performance of a parallel algorithm depends not only on input size but also on the architecture of the parallel computer, the number of processors, and the interconnection network. In this chapter, different types of parallel architectures and interconnection networks are discussed before actually implementing the parallel algorithm of encryption. At the end of this paper, some sample input/output are shown varying the key size, number of rounds and the number of processors to verify the correctness of parallel algorithm. Finally, the run time complexity of the parallel algorithm is shown to measure the performance improvement of the parallel implementation over the serial implementation.

## VII PARALLEL ARCHITECTURE

### *A. SIMD Architecture*

SIMD (Single-Instruction Stream Multiple-Data Stream) [19] architectures are essential in the parallel world of computers. In SIMD architectures, several processing elements are supervised by one control unit. All the processing units receive the same instruction from the control unit but operate on different data sets, which come from different data flows, meaning that they execute programs in a lockstep mode, in which each processing element has its own data stream. There are two types of SIMD architectures: the True SIMD and the Pipelined SIMD. Each has its own advantages and disadvantages but their common attribute is superior ability to manipulate vectors.

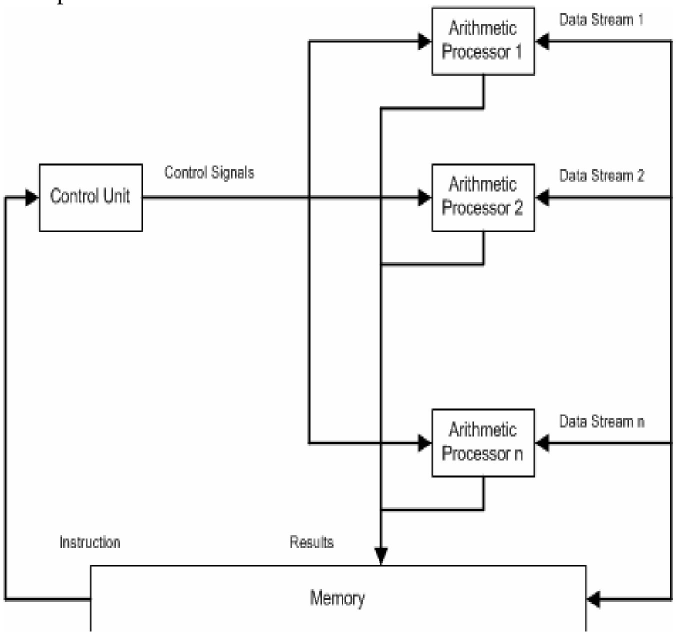

**Figure 5.1 Model of SIMD architecture**

### *B. MIMD Architecture*

Multiple instruction stream, multiple data stream (MIMD) [20] machines have a number of processors that function asynchronously and independently. At any time, different processors may be executing different instructions on different pieces of data. MIMD architectures may be used in a number of application areas such as computer-aided design/computer-aided manufacturing, simulation, modeling, and as communication switches. MIMD machines can be of either shared memory or distributed memory categories. These classifications are based on how MIMD
processors access memory. Shared memory machines may be of the bus-based, extended, or hierarchical type. Distributed memory machines may have hypercube or mesh interconnection schemes.

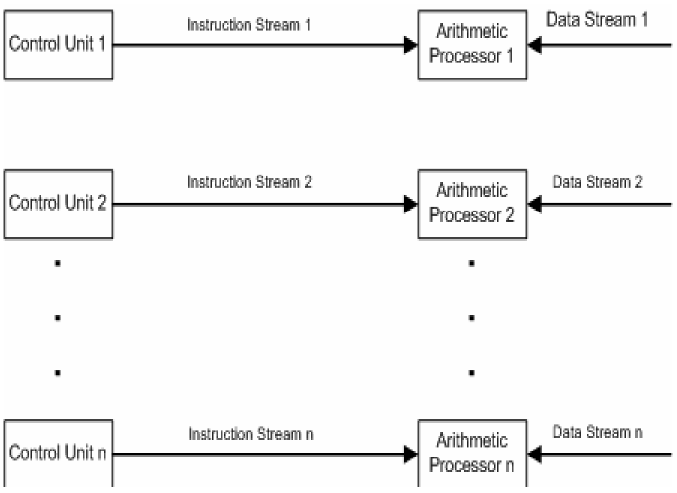

**Figure 5.2 Model of MISD architecture**

## VIII ALGORITHM FOR PARALLEL IMPLEMENTATION OF AES

There are two major components of parallel algorithm design. The first one is the identification and specification of the overall problem as a set of tasks that can be performed concurrently. The second is the mapping of these tasks onto different processors so that the overall communication overhead is minimized. The first component specifies concurrency, and the second one specifies data locality. The performance of an algorithm on a parallel architecture depends on both. Concurrency is necessary to keep the processors busy. Locality is important because it minimizes communication overhead. Ideally, a parallel algorithm should have maximum concurrency and locality. However, for most algorithms, there is a tradeoff. An algorithm that has more concurrency often has less locality.

To implement the AES algorithm in parallel, data blocks (Figure 5.3) and a key are distributed among the available processors. Each processor will encrypt different data blocks using the same key. For example, in order to encrypt **n** number of data blocks with p processors, n/p data blocks will be encrypted by each processor. As each processor has its own data blocks and a key (increases data locality), all the 10/12/14 rounds (consists of four transformations) will be executed by each processor for encrypting each data block. After encrypting all the data blocks of each processor, the encrypted data will be merged (Figure 5.4) in tree structure and return back to the main processor. For example, if there are four processors working in parallel, processor P1 will send its encrypted data to P0 and P0 will merge its encrypted data with P1; processor P3 will send its encrypted data to P2, and P2 will merge its encrypted data with P3. Finally processor P2 will send its (P2 & P3) encrypted data to P0 and P0 will merge its (P0 & P1) encrypted data with P2. This technique of merging and returning data to the main processor will increase the concurrency and reduce the idle time of each processor.

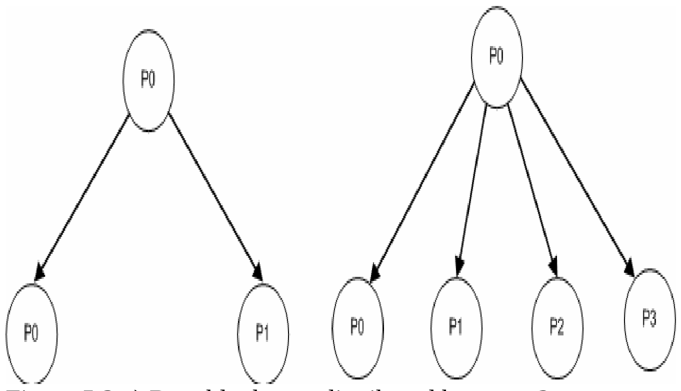

Figure 5.3 a) Data blocks are distributed between 2 processors

**b) Data blocks are distributed among 4 processors[14]**

Decryption is done in the same way as the encryption. After decrypting the data blocks, the plain texts are merged and return back to the main processor in the same way as described above.

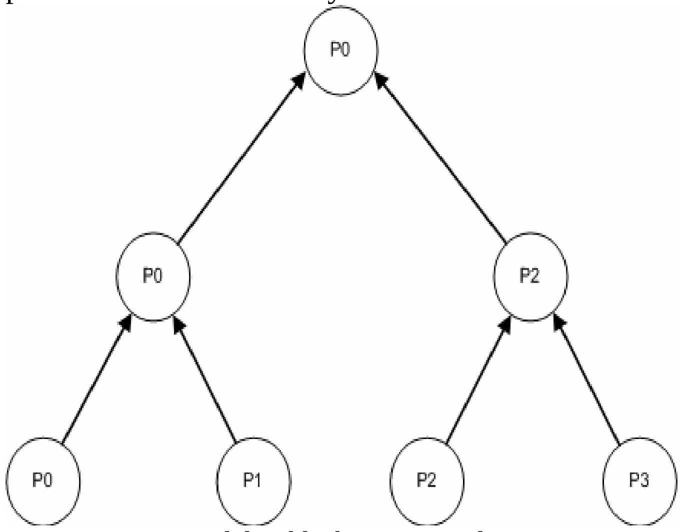

Figure 5.4 Encrypted data blocks are merged in tree structure

## IX THE OVERALL PARALLEL ALGORITHM OF AES CIPHER IS DESCRIBED BELOW

Constant: ArraySize = 160 ; int Nb = 4;
int Nr = 10, 12, or 14; // rounds, for Nk = 4, 6, or 8

Inputs: int nProcessors = 2/4/8/16 processors

int tNumberOfBlocks // number of blocks to be encrypted

unsigned char key[16] // key for encrypting data

int k = 0;

array w of 4*Nb*(Nr+1) bytes // expanded key

## X SAMPLE INPUT/OUTPUT

```
128-bit data, 128-bit Key
2 processors, each processor processes 4 data blocks
Encrypting . . .
Processor 0:
Plaintext1 32 43 F6 A8 88 5A 30 8D 31 31 98 A2 E0 37 07 34
Plaintext2 38 21 1A 00 0B 23 DE 93 F7 B6 65 7D F9 AE C4
D1
Plaintext3 AF DA 94 A5 E5 3C A1 25 B0 39 D3 58 0 CE BF
CA
Plaintext4 8E 9C 32 1E 84 47 CD BC 9B 67 7E B9 B6 23 5E
1A
Key 2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C
Ciphertext1 39 25 84 1D 02 DC 09 FB DC 11 85 97 19 6A 0B
32
Ciphertext2 E3 97 D6 93 C8 82 E9 DF 8A CD 3D 35 2A 20
0A 47
Ciphertext3 71 FB 5E D7 3E D0 76 AE C5 A1 89 14 86 70 16
3F
Ciphertext4 DB CA D3 9F C2 FC B6 EF F5 1B 60 39 53 1B
2B 24
Processor 1:
Plaintext1 D1 5B 5F 58 91 EA 82 C0 1F 28 4B 1A 37 B3 73
89
Plaintext2 3F 91 38 5A E1 F3 7B 9C 3D C2 AA 7B 9B F2 7C
23
Plaintext3 7C 70 DC B2 0F CC CA 20 5F 48 4F E7 43 34 07
88
Plaintext4 46 BA 06 15 41 1F 0F 96 30 46 06 AA 3E 76 A8
72
Key 2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C
Ciphertext1 B4 FD 4B E9 64 FF 4F 80 84 59 24 88 0F 21 54
F5
Ciphertext2 23 20 B7 96 CC 27 7A 91 E4 CC 1D 48 D4 75
3C 44
Ciphertext3 BC BD C4 72 EE F1 A7 9F 51 FF C3 2A E7 B1
52 7C
Ciphertext4 0F E9 FB 87 42 0F AA DD 0C C6 9C E1 40 F5
8B E4
Decrypting . . .
Processor 0:
Ciphertext1 39 25 84 1D 02 DC 09 FB DC 11 85 97 19 6A 0B
32
Ciphertext2 E3 97 D6 93 C8 82 E9 DF 8A CD 3D 35 2A 20
0A 47
Ciphertext3 71 FB 5E D7 3E D0 76 AE C5 A1 89 14 86 70 16
3F
Ciphertext4 DB CA D3 9F C2 FC B6 EF F5 1B 60 39 53 1B
2B 24
Key 2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C
Plaintext1 32 43 F6 A8 88 5A 30 8D 31 31 98 A2 E0 37 07 34
Plaintext2 38 21 1A 00 0B 23 DE 93 F7 B6 65 7D F9 AE C4
D1
Plaintext3 AF DA 94 A5 E5 3C A1 25 B0 39 D3 58 00 CE
BF CA
```

```
Plaintext4 8E 9C 32 1E 84 47 CD BC 9B 67 7E B9 B6 23 5E
1A
Processor 1:
Ciphertext1 B4 FD 4B E9 64 FF 4F 80 84 59 24 88 0F 21 54
F5
Ciphertext2 23 20 B7 96 CC 27 7A 91 E4 CC 1D 48 D4 75
3C 44
Ciphertext3 BC BD C4 72 EE F1 A7 9F 51 FF C3 2A E7 B1
52 7C
Ciphertext4 0F E9 FB 87 42 0F AA DD 0C C6 9C E1 40 F5
8B E4
Key 2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C
Plaintext1 D1 5B 5F 58 91 EA 82 C0 1F 28 4B 1A 37 B3 73
89
Plaintext2 3F 91 38 5A E1 F3 7B 9C 3D C2 AA 7B 9B F2 7C
23
Plaintext3 7C 70 DC B2 F CC CA 20 5F 48 4F E7 43 34 7 88
Plaintext4 46 BA 6 15 41 1F F 96 30 46 6 AA 3E 76 A8 72
128-bit data, 192-bit Key
2 processors, each processor processes 4 data blocks
Encrypting . . .
Processor 0:
Plaintext1 46 B7 15 DA 1A 1B 88 E4 15 23 C4 75 D7 C0 85
A5
Plaintext2 27 6B F4 4E 8B 66 D6 6F 15 44 04 CE 7F AB 1B
E2
Plaintext3 7B 59 74 EF 44 65 57 73 79 C7 6B C1 2B 1B 71
EC
Plaintext4 35 A2 32 E6 CB 8A B6 E9 56 6F B7 16 7F 21 55
5D
Key E9 55 A9 D5 0A B2 91 60 E0 0B 3B 86 41 A8 5C 77
22 5C 7A AA AE 54 FF 19
Ciphertext1 FE 90 46 41 DB FC 64 28 D7 52 97 48 25 77 C1
92
Ciphertext2 EF AD 0A 1D CA 8D F1 97 CA 4C 26 E3 B0 9D
A3 D0
Ciphertext3 62 7B 8C DE B8 2D 32 28 52 11 A0 9A EE BE
AA 69
Ciphertext4 F1 6C AC BB C6 30 7A 6E BA AD A3 FA C9
1A 24 11
Processor 1:
Plaintext1 8A A8 C 9C E6 85 E2 B 02 3B E5 D7 62 0E D4
0D
Plaintext2 F1 0B 1D BA 99 48 07 51 11 6D 31 4C F6 73 3B
16
Plaintext3 1E AC C3 29 29 4 93 73 58 86 18 FF A3 22 17 D1
Plaintext4 05 AD 9B 12 1B 29 32 6D ED 47 58 B8 0C 2A 34
D7
Key E9 55 A9 D5 A B2 91 60 E0 0B 3B 86 41 A8 5C 77
22 5C 7A AA AE 54 FF 19
Ciphertext1 EF 26 42 A5 62 DC 10 77 46 B0 14 13 A2 D2 69
86
Ciphertext2 C8 AC E9 E3 6E D8 9B 93 B9 B0 9D 63 05 F0
E6 14
Ciphertext3 8A 76 86 D3 E4 F2 85 D1 78 BA 52 0A 6C 6E
83 BE
Ciphertext4 32 CE E2 54 8E 3A F2 E6 13 66 A8 C3 FE 26 6B
6B
Decrypting . . .
Processor 0:
Ciphertext1 FE 90 46 41 DB FC 64 28 D7 52 97 48 25 77 C1
92
Ciphertext2 EF AD 0A 1D CA 8D F1 97 CA 4C 26 E3 B0 9D
A3 D0
Ciphertext3 62 7B 8C DE B8 2D 32 28 52 11 A0 9A EE BE
AA 69
Ciphertext4 F1 6C AC BB C6 30 7A 6E BA AD A3 FA C9
1A 24 11
Key E9 55 A9 D5 A B2 91 60 E0 B 3B 86 41 A8 5C 77
22 5C 7A AA AE 54 FF 19
Plaintext1 46 B7 15 DA 1A 1B 88 E4 15 23 C4 75 D7 C0 85
A5
Plaintext2 27 6B F4 4E 8B 66 D6 6F 15 44 4 CE 7F AB 1B
E2
Plaintext3 7B 59 74 EF 44 65 57 73 79 C7 6B C1 2B 1B 71
EC
Plaintext4 35 A2 32 E6 CB 8A B6 E9 56 6F B7 16 7F 21 55
5D
Processor 1:
Ciphertext1 EF 26 42 A5 62 DC 10 77 46 B0 14 13 A2 D2 69
86
Ciphertext2 C8 AC E9 E3 6E D8 9B 93 B9 B0 9D 63 05 F0
E6 14
Ciphertext3 8A 76 86 D3 E4 F2 85 D1 78 BA 52 0A 6C 6E
83 BE
Ciphertext4 32 CE E2 54 8E 3A F2 E6 13 66 A8 C3 FE 26 6B
6B
Key E9 55 A9 D5 A B2 91 60 E0 B 3B 86 41 A8 5C 77
22 5C 7A AA AE 54 FF 19
```

## XI RUN TIME COMPLEXITY OF THE PARALLEL IMPLEMENTATION

Time complexity is the most important measure of the performance of a parallel algorithm, since the primary motivation for parallel computation is to achieve a speedup in the computation. Parallel algorithms are executed by a set of processors and usually require inter-processor data transfers to complete execution successfully. The time complexity of a parallel algorithm to solve a problem of size n is a function T(n,p) which is the maximum time that elapses between the start of the algorithm's execution by one processor and its termination by one or more processors with regard to any arbitrary input. There are two different kinds of operations associated with parallel algorithms[15]:

- Elementary operation
- Data routing operation

**Elementary operation** is an arithmetic or logical operation performed locally by a processor. **Data routing operations** refer to the routing of data among processors for exchanging the information. The time complexity of a parallel algorithm is

determined by counting both elementary steps and data routing steps.
It is also important to evaluate the speedup factor and efficiency of the algorithm. All of them are described below:

### *A. Run Time*

The serial run time of a program is the time elapsed between the beginning and the end of its execution on a sequential computer. The parallel run time is the time that elapses from the moment that a parallel computation starts to the moment that the last processor finishes execution. The serial and parallel run time is denoted by TS and TP respectively.

### *B. Speed-up*

When evaluating a parallel system, it is often important to know how much performance gain is achieved by parallelizing a given application over a sequential implementation. Speed-up is defined as the ratio of the time taken to solve a problem on a single processor to the time required to solve the same problem on a parallel computer with p identical processors. The speedup is denoted by the symbol S. Therefore, S = TS / TP Formally, the speedup S is defined as the ratio of the serial run time of the **best sequential algorithm** for solving a problem to the time taken by the parallel algorithm to solve the same problem on P processors [16].

### *C. Efficiency*

Efficiency is defined as the Speed-up with N processors divided by the number of processors N. Conceptually, the efficiency of the algorithm measures how well all N processors are being used when the algorithm is computed in parallel. An efficiency of 100 percent means that all of the processors are being fully used all the time. Efficiency is denoted by E. Therefore,
E = S / P = TS / P TP

## XII CONCLUDING REMARKS

From the above discussions, it is found that the efficiency of parallel AES ranges from 0.75 to 0.82 and the speedup is less than the number of processors used in the parallel implementation. In practice, it is not possible to achieve speedup equal to P and efficiency equal to 1, because while executing a parallel algorithm, the processors cannot devote 100 percent of their time to the computations of the algorithm. The time to transfer data between processors is equally the most significant source of parallel processing overhead.

Another reason is load unbalancing i.e. not all the processors are equally busy while executing a parallel program. If different processors have different workloads, some processors may be idle during part of the time that others are working on the problem. It is very difficult to develop any parallel algorithm where each processor will work equal amount of time. After implementing the encryption algorithm in parallel, it is found that the performance of algorithm increases significantly as the number of processor increases. It is not possible to get the speedup factor equal to P (number of processor), as some parallel processing overhead is also occurred during the implementation of AES in parallel.

## XIII SUGGESTION FOR FUTURE WORK

As the performance of any parallel implementation depends on the parallel architectures and the interconnection networks, the same parallel implementation can give different performance on different architectures and interconnection networks. So, there is a good opportunity to work on some other architectures and interconnection networks and find out which architecture and interconnection network will be suitable for parallel implementation of encryption.

## REFERANCES

[1] Menezes, A. and Vanstone, S. "Handbook of Applied Cryptography", CRC Press, Inc. 1996

[2] National Bureau of Standards, NBS FIPS PUB 46, "Data Encryption Standard,", U.S. Department of Commerce, January 1977.

[3] Coppersmith, D. "The Data Encryption Standard (DES) and Its Strength Against Attacks." IBM Journal of Research and Development, May 1994.

[4] Diffie, W. and Hellman, M. "Multiuser Cryptographic Techniques" proceedings of AFIPS National Computer Conference, 1976, 109-112

[5] Korner. T. "The Pleasures of Counting", Cambridge, England, Cambridge University Press, 1996

[6] Khan, D. "The Codebreakers: The Story of Secret Writing." New York, 1996

[7] Schneier, B. "Applied Cryptgraphy." New York: Wiley, 1996.

[8] Stallings, W. "Cryptography and Network Security: Principles and Practices." Third Edition, Pearson Education, Inc. 2003.

[9] Rivest, R., Shamir, A. and Adleman, L. "A Method for Obtaining Digital Signature and Public-Key Cryptosystems." Communication of the ACM, 21, 2, Feb 1978, 120-126.

[10] RSA Laboratories, "The RSA Laboratories Secret-key Challenge: Cryptographic Challenges" , www.rsasecurity.com/rsalabs/node.asp

[11] Daemon, J. and Rijmen, V. "The Rijndael Block Cipher: AES Proposal", NIST, Version 2, March 1999.

[12] Daemon, J., and Rijmen, V. "Rijndael: The Advanced Encryption Standard." Dr. Dobb's Journal, 26, 3, March 2001, 137-139.

[13] Daemon, J., and Rijmen, V. "The Design of Rijndael: The Wide Trail Strategy Explained." New York, Springer – Verlag, 2000.

[14] National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information, CNSS Policy No. 1, Fact Sheet No. 1, June 2003, www.nstissc.gov/Assets/pdf/fact%20sheet.pdf

[15] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. In Proc. 2nd AES candidate conference, pp 15–34, NIST, 1999, www.macfergus.com/pub/icrijndael.html

[16] Lidl, R., and Niederreiter, H. "Introduction to finite fields and their applications" Cambridge University Press, 1986.