2018

Towards Designing Energy Efficient Symmetric Key Protocols

Sai Raghu Talluri
University of North Florida, n00926109@ospreys.unf.edu

TOWARDS DESIGNING ENERGY EFFICIENT SYMMETRIC KEY PROTOCOLS

by

S. Raghu Talluri

A thesis submitted to the
School of Computing
in partial fulfillment of the requirements for the degree of

Master of Science in Computer and Information Sciences

UNIVERSITY OF NORTH FLORIDA
SCHOOL OF COMPUTING

August, 2018

The thesis “Towards Designing Energy Efficient Symmetric Key Protocols” submitted by S. Raghu Talluri in partial fulfillment of the requirements for the degree of Master of Science in Computing and Information Sciences has been

Approved by the thesis committee: Date:

Dr. Swapnoneel Roy
Thesis Advisor and Committee Chairperson

Dr. Asai Asaithambi

Dr. Roger E. Eggen

Accepted for the School of Computing:

Dr. Sherif Elfayoumy
Director of the School

Accepted for the College of Computing, Engineering, and Construction:

Dr. William F. Klostermeyer
Dean of the College

Accepted for the University:

Dr. John Kantner
Dean of the Graduate School

ACKNOWLEGEMENTS

Firstly, I would like to express my sincere gratitude to my advisor Dr. Swapnoneel Roy for his continuous support of my Masters study and related research, and for his patience, motivation, and immense knowledge. I could not have imagined having a better advisor and mentor. He was so understanding and always ready to help with my research, internships and for career guidance. I think meeting him is one of the best things that has ever happened to me. I would also like to thank my committee members, Dr. Roger Eggen and Dr Asai Asaithambi, for providing insightful comments during my prospectus presentation. I would like to especially thank Dr. Asai Asaithambi for being so patient in guiding and preparing me for my thesis presentation and Dr. Roger Eggen for academic support and advice for my internship. I also want to thank all my professors and Ms. Shawn Broderick for their help and encouragement.

Last but not the least, I would like to thank my parents and my uncle for making all this possible.

CONTENTS

FIGURES

TABLES

ABSTRACT

Energy consumption by various modern symmetric key encryption protocols (DES, 3-DES, AES and, Blowfish) is studied from an *algorithmic* perspective. The work is directed towards redesigning or modifying the underlying algorithms for these protocols to make them consume less energy than they currently do. This research takes the approach of reducing energy consumption by parallelizing the consecutive memory accesses of symmetric key encryption algorithms. To achieve parallelization, an existing energy complexity model is applied to symmetric key encryption algorithms. Inspired by the popular DDR3 architecture, the model assumes that main memory is divided into multiple *banks*, each of which can store multiple blocks. Each block in a bank can only be accessed from a cache of its own, that can hold exactly one block. However all the caches from different banks can be accessed simultaneously. In this research, experiments are conducted to measure the difference in energy consumption by varying the level of parallelization, i.e. variations of, number of banks that can be accessed in parallel. The experimental results show that the higher the level of parallelism, smaller is the energy consumption.

CHAPTER 1

Introduction

1.1 Security and Energy Efficiency: The Balance

The importance of data centers has been growing rapidly, because they facilitate infrastructure for a wide range of applications and services including social and business networking, internet marketing, electronic banking and insurance, web mail, storage and high performance computing. All of these applications involve storing, manipulating and transmitting sensitive data, which has led to increased concern about securing the data. These applications run on battery-driven devices such as laptops, personal digital assistants and smartphones, making energy consumption an important factor as well. Moreover, the importance of considering energy and security factors in developing technological applications has been growing with the advent of next generation technologies such as ubiquitous computing and the “internet of things" in which entities such as thermostats, homes, and cities are connected to the internet through electronic sensors. The electronic sensors or sensor nodes are powered by batteries with finite life, making energy consumption a crucial factor in developing such applications.

Also security of cryptographic protocols has become an integral part of the design of technological applications. Cryptographic protocols transform the data into scrambled form so that an adversary or unauthorized person does not get access to the data. Cryptographic protocols that provide a higher level of security require extensive use of modular exponentiation [36]. Since modular exponentiation is computationally intensive and time consuming, the higher the level of security, the higher the energy consumption. The energy consumption and the resulting draining of battery lead to a major concern for portable devices if security needs to be enforced [36]. Moreover, next generation technological applications that use electronic sensors will require even more energy to secure the data using cryptographic protocols. One study [25] showed that encryption increased energy consumption significantly. A battery powered sensor node consumed 21 Millijoule(mJ) of energy for transmitting 1024 bits of data. But when encryption was used, it consumed 42 mJ. Thus energy consumption doubled when encryption was used, and the battery was depleted twice as fast when compared to no encryption being used. Therefore the impact of cryptographic protocols on energy consumption warrants further investigation. A considerable amount of work has been done to reduce energy consumption in network communication and security protocols at the hardware, virtual machine, operating system and system software levels [18] [22] [17]. However, no work has been reported on modifying encryption protocols at algorithmic level, to reduce energy consumption.

1.1.1 Problem Statement

The research in this thesis takes the approach of reducing energy consumption by parallelizing the consecutive memory accesses of symmetric key encryption algorithms, so that data is read in parallel in blocks to achieve the required level of parallelization. To achieve parallelization, the energy complexity model of Roy *et al.* [29] is applied to symmetric key encryption algorithms such as Data Encryption Standard (DES), Triple Data Encryption Standard (3-DES), Blowfish, and Advanced Encryption Standard (AES). We investigate the impact of applying the model on these four algorithms.

Problem: Does any difference in energy consumption result from applying the energy complexity model of Roy *et al.* [29] to the symmetric key encryption algorithms?

1.2 Contribution of this Thesis

There are three key contributions of this research. The first is a detailed experimental evaluation of energy consumption by four symmetric key encryption protocols, DES, 3-DES, AES and Blowfish. Secondly, energy efficient security protocols are developed. The experimental results validate that there is difference in energy consumption by applying the energy complexity model. A generic technique to achieve any desired degree of memory parallelism for a given symmetric block cipher is presented, and the main parameters that impact energy consumption of the security protocols are identified. The third contribution is a set of recommendations for energy-aware security protocol design.

1.3 Organization of this Thesis

In Chapter 2, background information regarding previous work done on reducing energy consumption in security protocols is presented. Chapter 3 describes the energy complexity model [29] and the methodology by which we applied the model to the symmetric key encryption protocols. Chapter 4 describes and analyzes the results obtained. Finally, contributions are summarized and possible future work is described in Chapter 5.

CHAPTER 2

Background

2.1 Symmetric Key Encryption Algorithms

Encryption is the process of converting a humanly readable message (original message) into a scrambled message; decryption is the process of reconstructing the original message from the scrambled message.

- **Plaintext** The original message that is supposed to be encrypted and is given as input to the encryption algorithm.
- **Ciphertext** The scrambled message (original message encrypted).
- **Key** The key is a rule used to convert plain text into cipher text

Encryption algorithm is also referred to as a cipher which means secret or disguised way of writing. Symmetric key encryption is a kind of encryption in which only one key is used for both encryption and decryption. It is also called conventional encryption or secret-key encryption.

- **Secret key** Input to the encryption algorithm that is independent of plaintext and ciphertext. The output produced by the encryption algorithm depends on the secret key.

- **Encryption algorithm** Performs certain substitutions and transformations on the plaintext, using the secret key.

- **Decryption algorithm** The inverse of the encryption algorithm. It takes the secret key and ciphertext as input and regenerates the original message.

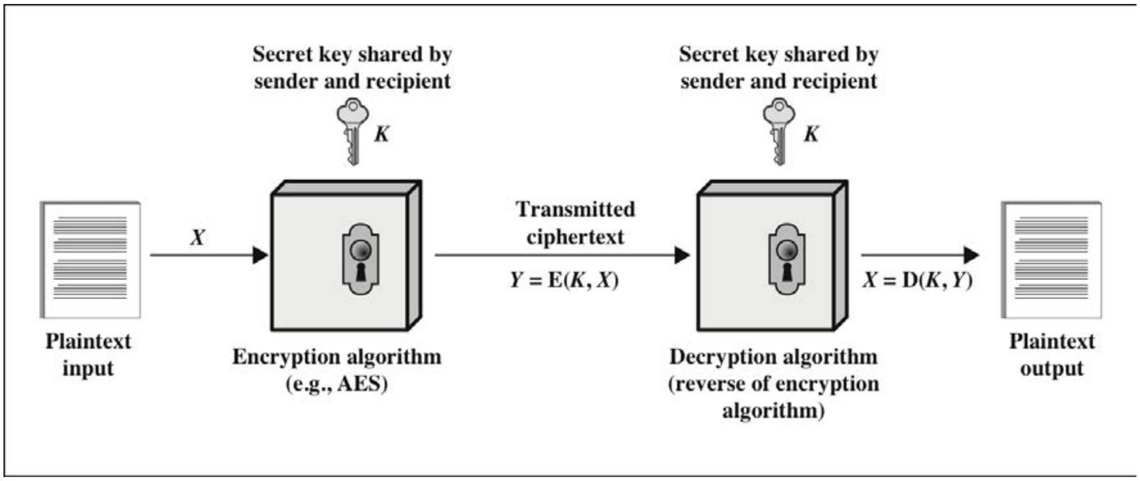

Secret key shared by sender and recipient
K
Secret key shared by sender and recipient
K
X
Transmitted ciphertext
Y = E(K, X)
X = D(K, Y)
Plaintext input
Encryption algorithm (e.g., AES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

Figure 2.1: Symmetric Key Encryption.

As shown in Fig 2.1, symmetric ciphers take plaintext and encrypt it with the encryption algorithm, using the secret key, to produce the resulting ciphertext. The decryption algorithm takes the ciphertext and the same secret key to generate the original plaintext. The term symmetric refers to the fact that the decryption process is identical to encryption in that it uses the same secret key and algorithm. In contrast, asymmetric key algorithms use two keys, one for encryption and one for decryption, and the encryption and decryption algorithms are different. There are two kinds of symmetric key ciphers, stream ciphers and block ciphers.

Stream ciphers encrypt one bit at a time; the plaintext is divided into bits/bytes and then each bit/byte is encrypted and concatenated. In stream ciphers, the key size is the same as the size of plaintext.

Block ciphers encrypt in blocks; the plaintext is divided into blocks of size n (determined by the algorithm), and the blocks are encrypted using the secret key and then concatenated. Decryption follows the same procedure, that is, it divides the ciphertext into blocks, decrypts them with the same key and concatenates yielding the plaintext. Block ciphers have fixed length block sizes and key sizes, independent of the size of the plaintext. All of the algorithms considered in this work are block ciphers.

2.2 Previous Work

The importance of designing energy efficient security protocols has been growing rapidly. Some work has been done to reduce energy consumption in security protocols. Energy consumption of symmetric key and asymmetric key algorithms has been studied in wireless sensor networks [3, 19, 25]. Carman et al. (2000) suggested energy efficient techniques to ensure security in a distributed wireless sensor network. They suggested the use of hybrid encryption protocols and devised hybrid key management protocols. However, the energy savings were specific to sensor networks and based entirely on their topology. Hodgat and Verbauwhede [19] compared symmetric key encryption on a non-ad hoc network with public key encryption on an ad hoc network. In particular, they showed that AES symmetric key encryption on a non-ad hoc network is 10 to 100 times more energy efficient than Rivest Shamir Adleman (RSA) and Elliptic Curve Cryptography (the predominant public key encryption protocols). Karri and Mishra [21] described techniques to reduce the power consumption of a secure wireless session without compromising the security of the session. They showed that compressing the message and making the compression size equal to the cache size would give good energy savings. A framework has been proposed to design energy efficient security protocols in [26]. The framework provides guidelines for the implementation and execution of security protocols. A detailed analysis of energy consumption of symmetric, asymmetric and hash algorithms was performed and the associated bottlenecks identified in [25].

2.2.1 Gap in the Literature

All of the previous published work concerns developing custom communication protocols for wireless sensor devices and analyzing the energy consumption of encryption algorithms and hashes. The literature review has not revealed any work done on reducing the energy consumption of existing security protocols or analyzing the energy consumption of the protocols from an algorithmic perspective. The research in this thesis focuses on reducing energy consumption by modifying the existing algorithms and by parallelizing the consequent memory accesses.

2.3 Encryption Algorithms considered

Data Encryption Standard (DES), Triple Data Encryption Standard (3-DES), Blowfish, and Advanced Encryption Standard (AES) are the protocols considered for the research in this thesis. Table 2.1 lists the symmetric key algorithms considered along with the sizes of their input blocks, keys, and any known vulnerabilities.

Protocol	**Input Block Size (bits)**	**Key Size (bits)**	**Vulnerabilities**
DES	64	56	Bruteforcing over keys, Cryptanalysis
3-DES	64	56, 112, or 168	Computationally inefficient
AES	128	128, 192, or 256	None
Blowfish	64	32–448	Weakness in key selection

Table 2.1: The Symmetric Key Algorithms Considered.

Sections 2.3.1, 2.3.1, 2.3.3 give a brief introduction to these algorithms.

2.3.1 DES and 3-DES

The DES symmetric key algorithm was developed by IBM in the early 1970s to encrypt electronic data. It was accepted as the Federal Information Processing Standard in 1977 [13] and is one of the most extensively used key ciphers [2]. The DES algorithm is being researched extensively for the purpose of encrypting data in the cloud, where data is stored and accessed from the network and hence there is a high need to keep data secure [2, 20, 14]. Ongoing research aims at optimizing the DES algorithm, using techniques like dynamic key generation [16].

3-DES is an enhanced version of DES in which each block is encrypted three times using DES, making it more secure. It is used by the electronic payment industry to protect data on smart cards and also in some online transactions [8]. Microsoft applications such as OneNote, Outlook 2010, and Microsoft System Center Configuration Manager 2012 use 3-DES to password-protect user content and data [9]. 3-DES is also used in the cloud, and research has shown that 3-DES encrypts and decrypts faster than RSA [20], a widely used cryptographic algorithm.

2.3.2 AES

Established by the National Institute of Standards and Technology in 2001, AES is a specification for encrypting electronic data. It is used worldwide and is an open source software [5]. AES has applications in a wide variety of fields such as steganography, state machine design, radar parameters, VLSI architecture, and RFID [12, 23, 39, 33, 7, 40]. Since AES is open source, we cannot specifically point out all the places where it is used. However we can definitely say that it is widely used as any system that uses Secure Sockets Layer (SSL)/Transport Layer Security (TLS)[1] uses either AES or 3-DES for encryption [6]. Even though GSM phones do not use AES algorithms, every smartphone that engages in web traffic goes through HTTPS[2] and uses SSl/TLS [10], and hence employs either AES or 3-DES for encryption.

2.3.3 Blowfish

Blowfish is a symmetric key cipher designed by Bruce Schneier in 1993. There is no known successful attack on Blowfish provided that the number of rounds of encryption (derived from the key size) in the implementation is 14 or more [31]. There is no patent on Blowfish and it can be used freely in all countries [32].

[1] SSL and TLS are standard security mechanisms used to keep an internet connection secure and safeguard any sensitive data. SSL/TLS encrypts the data sent over the internet and uses symmetric key encryption to encrypt the data.

[2] Hyper Text Transfer Protocol Secure(HTTPS) appears on a web-page when the HTTP is secured using SSL/TLS

Blowfish is used for file and disk encryption, password management, archiving tools, backup software, database security and many more applications [31].

It can be clearly seen from subsections 2.3.1, 2.3.2, and 2.3.3 that the symmetric ciphers DES, 3-DES, AES and Blowfish are the most extensively used ciphers. These ciphers are used in cloud based services and wireless sensor nodes where battery life is very important. They are used on a large scale all over the world, so even the slightest reduction in energy consumption could save a significant amount of energy.

CHAPTER 3

Methodology

3.1 The Energy Complexity Model of Roy *et al.*

In order to redesign symmetric key algorithms for the purpose of reducing energy consumption, a memory model referred to as the energy complexity model [29] is employed in this study. Inspired by the popular DDR3 architecture [11], the model assumes that memory (RAM) is divided into P banks $(M_1, M_2, \cdots, M_P)$, each of which can store multiple blocks of size B. Each bank M_i has its own cache C_i that can hold exactly one block. A block can be accessed only when it is in the cache of the bank to which it belongs. Therefore only one block of a particular bank can be accessed at a particular time. But P blocks in P different memory banks can be accessed in parallel (Fig. 3.9). The difference in power consumption between the same number of blocks (n) accessed sequentially and all in parallel is not significant. But the authors of [29] discovered that the execution time of an algorithm when blocks are accessed in parallel is much lower than when blocks are accessed sequentially. The total energy consumption of an algorithm depends on the power consumed by the algorithm and its execution time. Therefore parallelizing block accesses leads to a reduction in energy consumption of any algorithm. In particular, the energy consumption $E(\mathcal{A})$ of an algorithm as derived by Roy *et al.*,

$$E(\mathcal{A}) \propto \frac{1}{k} \tag{3.1}$$

is inversely proportional to the parallelization factor k.

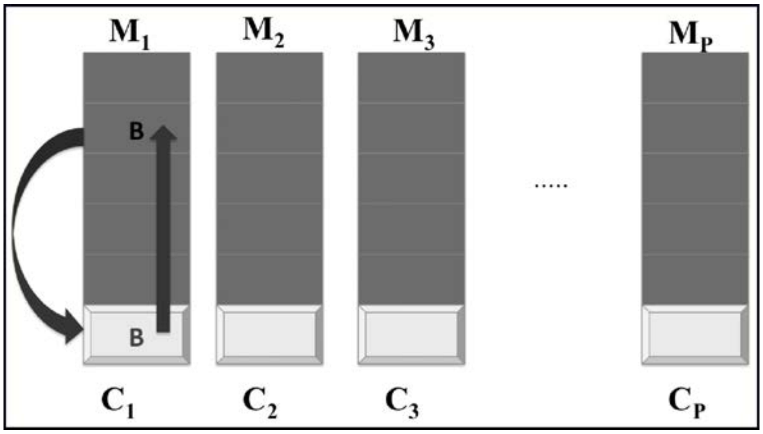


Figure 3.1: The Energy Complexity Model.

Hence, it is conjectured in this work, that parallelizing the memory accesses of blocks of any symmetric key algorithm, will reduce energy consumption of that algorithm. Therefore, by increasing the parallelization factor in symmetric key encryption algorithms, their energy consumption is reduced.

3.2 Application of the Energy Complexity Model to Symmetric Key Algorithms

Each of the protocols considered (Blowfish, DES, 3-DES, and AES) is a block cipher operating on fixed-length groups of bits, called blocks (of size b[1]). As discussed in Section 2.1, block ciphers encrypt in blocks. The plaintext is divided into blocks of size b (determined by the algorithm), and these blocks are encrypted using the secret key and then concatenated to produce the ciphertext (Fig 3.2).

Symmetric key encryption algorithms fit well into the energy complexity model because blocks in the cipher can be placed into the blocks in the energy model

[1] We use B to denote the size of a memory block, and b to denote the size of a plaintext/ciphertext block to differentiate between the two.

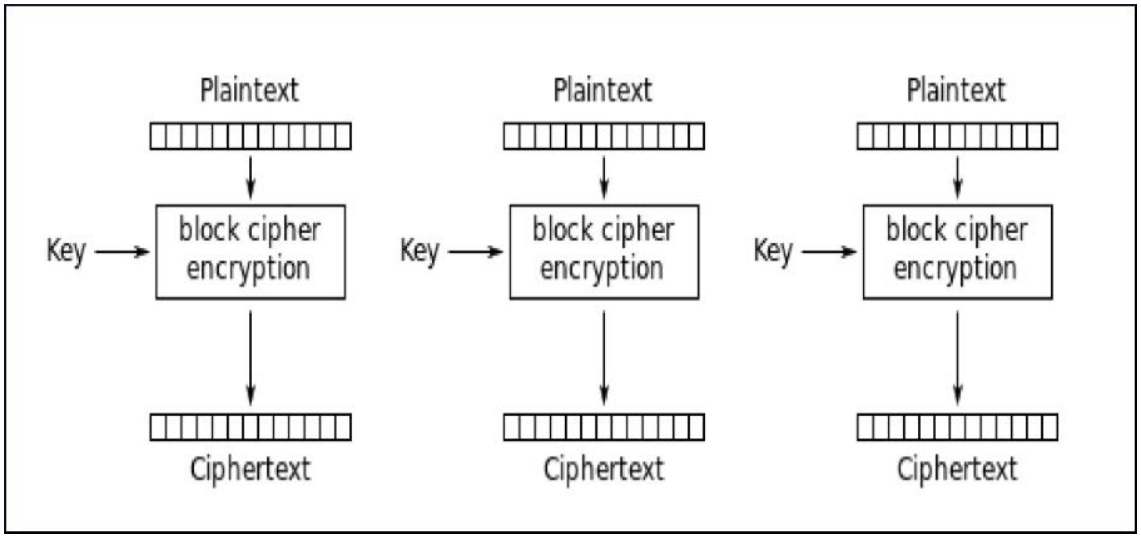


Figure 3.2: Block Ciphers.

as shown in Fig 3.3. To apply the energy complexity model to symmetric key algorithms, a function is defined and implemented to map blocks (of size b) of the algorithms to blocks (of size B) of memory banks. $B > b$ in this work[2].

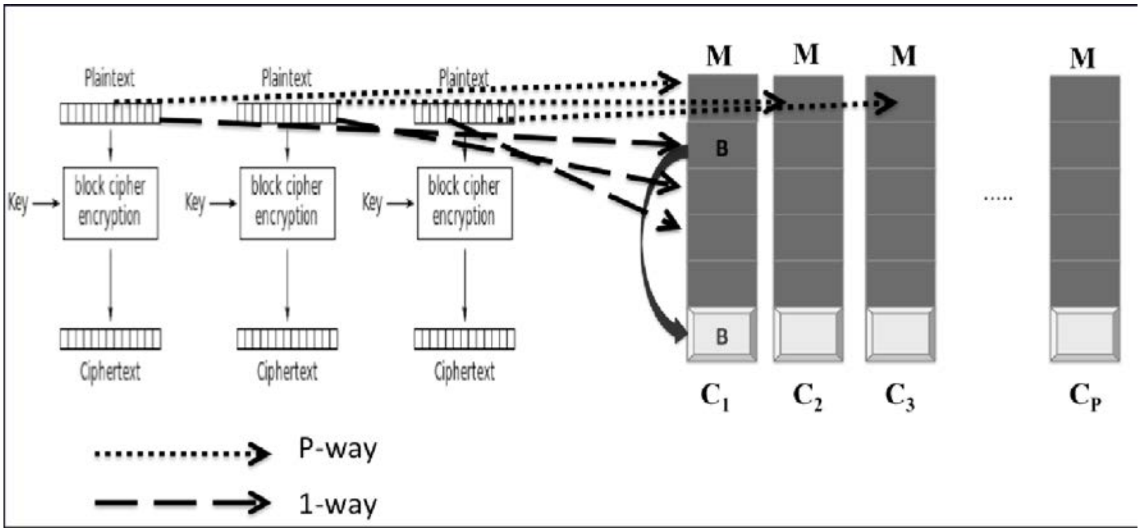


Figure 3.3: Implementing Parallelism for Symmetric Key Protocols

[2] In most cases B is significantly higher than b

3.2.1 An Illustrative Example

In this section we present an example on how the above mapping function works. For this example we assume a memory of four banks $P = 4$. As mentioned above, the block size b of the input (plaintext) is less than the block size B of the memory banks for all the ciphers we have considered in this work. Therefore each memory block will contain $\frac{B}{b} > 1$ plaintext blocks in it. But for simplicity of explanation, the author of this thesis does not differentiate between the two different types of blocks in the example illustrated below. The value of each parameter (P, B, b) will be specified for all four ciphers in subsequent sections.

For a given plaintext message input M, the algorithm divides M into blocks of B bytes[3] and processes each block for encryption to produce the ciphertext. A logical mapping is created, which ensures access to the blocks in P-way parallel fashion, where P ranges from 1 to 4. More specifically, when $P = 1$, consecutive blocks are clustered in a single bank (Fig. 3.4).

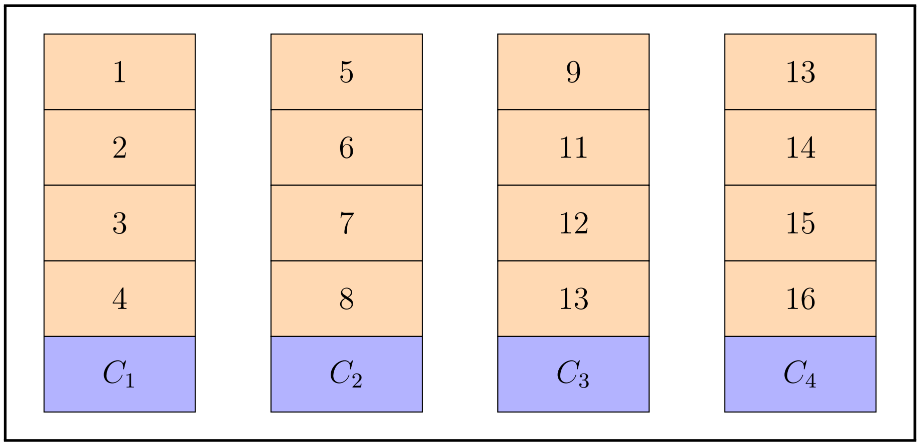


Figure 3.4: 1-Way Parallelism

[3] For this example, as mentioned above, we assume $B = b$.

For $P = 4$, consecutive blocks are evenly spread across all 4 banks (Fig. 3.5).

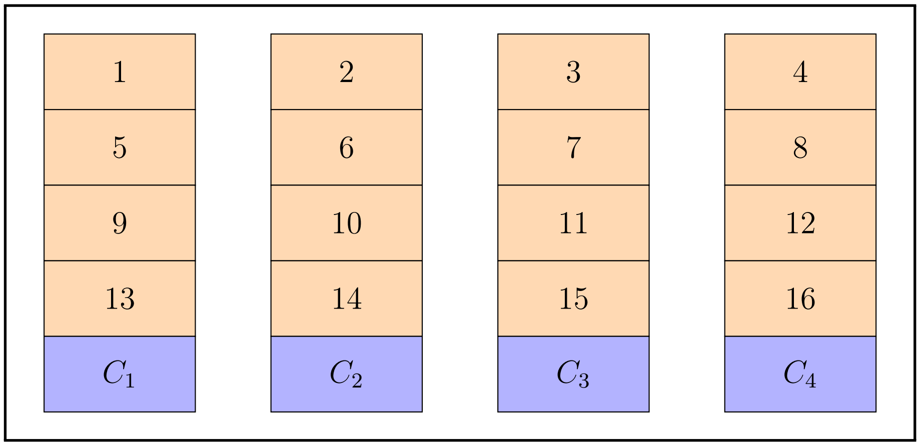


Figure 3.5: 4-Way Parallelism

3.2.2 Implementation of the Logical Mapping

The allocation of plaintext blocks to the blocks of the memory banks is done by the memory controller of the DDR3 RAM. The memory controller uses several strategies to allocate memory to an application that is not within the control of the end user [1]. Therefore in order to implement P-way allocation, in this work, P-way accesses to blocks have been programmed. Examples of different access patterns are illustrated under the assumption blocks are physically allocated as in Fig 3.6.

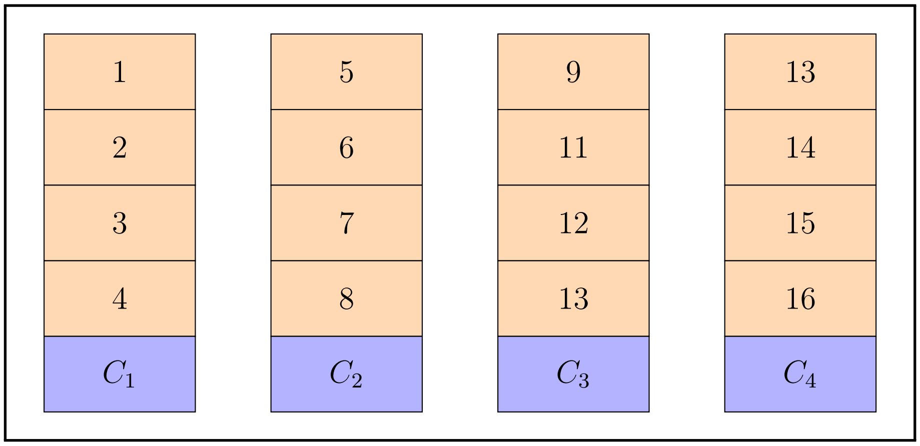


Figure 3.6: Physical Allocation of Blocks

With blocks allocated as shown in Fig 3.6 by the memory controller, a 1-Way access pattern over the blocks is shown in Fig 3.7.

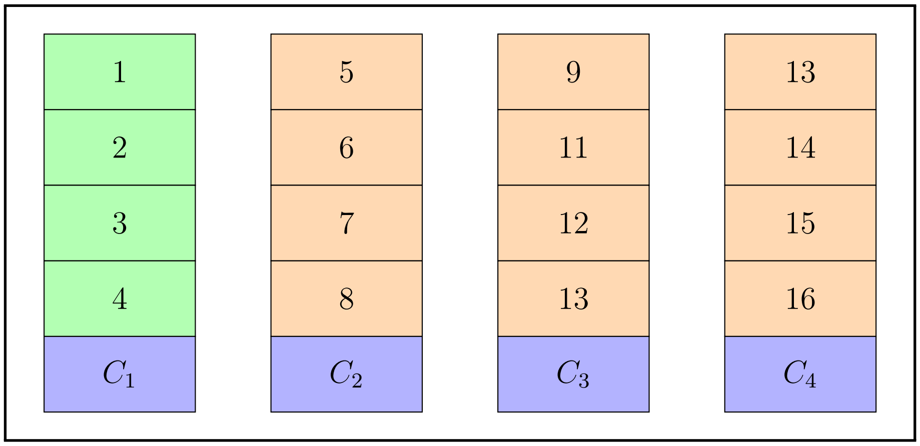


Figure 3.7: 1-Way Access Pattern

For $P = 1$ (1-way access pattern), all the consecutive accesses $\{1, 2, 3, 4\}$ (shown green) are clustered in a single bank (bank 1 of Fig. 3.7). A 4-way access pattern is defined over the same physical memory allocation in Fig. 3.8.

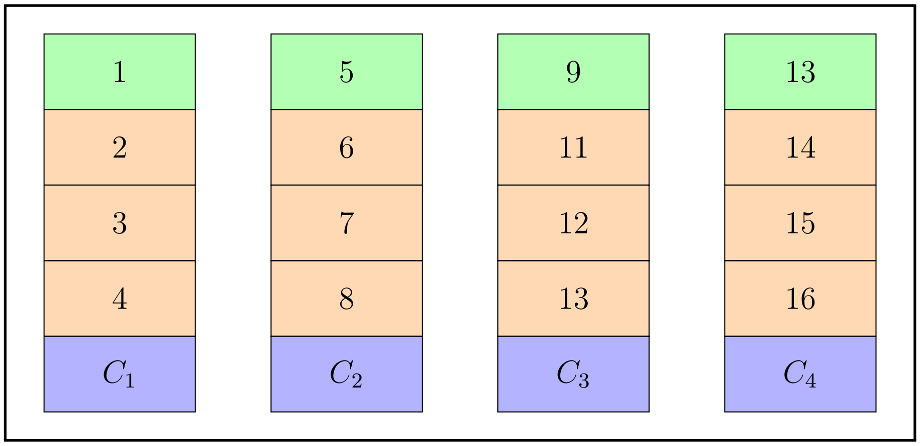


Figure 3.8: 4-Way Access Pattern

For $P = 4$ (4-way access), all the consecutive accesses $\{1, 5, 9, 13\}$ (again shown green) lie in different banks and can be accessed in parallel simultaneously (Fig. 3.8).

3.2.2.1 Logical Mapping

As mentioned above, the order of accesses is different for different levels of parallelization, i.e. for $(P = 1) \implies \{1, 2, 3, 4\}$, and for $(P = 4) \implies \{1, 5, 9, 13\}$. But the physical location of the input in the memory is static. To implement this, a different page table vector $\mathbf{T}$ is generated for each level of paralellization, which defines the ordering among the blocks to be accessed (Fig. 3.9). For 1-way, the page table vector $\mathbf{T}$ has indexes $\{1, 2, 3, 4, \cdots\}$ and for 4-way it has indexes $\{1, 5, 9, 13, \cdots\}$. A mapping function is then created to map the indexes of the page table vector $\mathbf{T}$ to the original physical location indexes of the input.

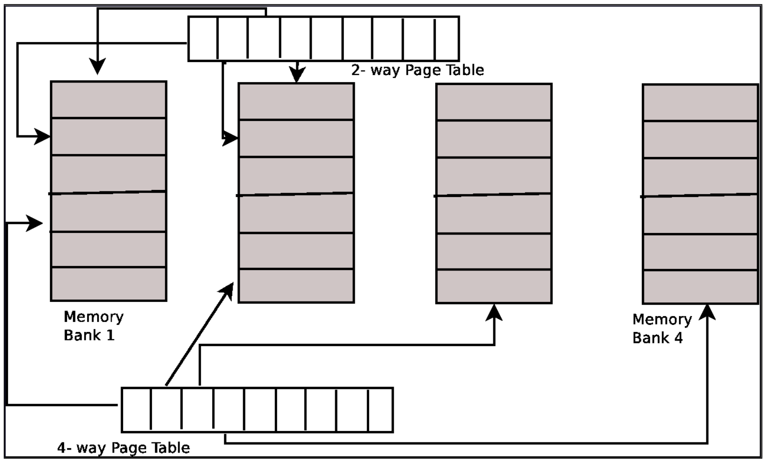


Figure 3.9: Memory Layout for $P = 4$

3.2.2.2 Ordering Among the Blocks

Algorithm 1 shows the function to create an ordering among the blocks. The ordering is based on the way we want to access the blocks (P-way would mean full parallel access). The page table is populated by picking blocks with jumps. For P-way access, jumps of P are selected that ensure the consecutive blocks accesses lie in P different banks. Going by the above example, for $P = 1$, jumps of 1 ensure that 4 consecutive blocks accesses lie in the same bank (bank 1 of Fig. 3.7). On the other hand, for $P = 4$, jumps of 4 ensures that 4 consecutive blocks access lie in 4 different banks (banks 1 through 4 of Fig. 3.8).

```
Input: Page table vector $\mathbf{V}$, jump amount $jump$.
$factor = 0$;
for $i = 0$ to $\frac{N}{B} - 1$ do
    if $i > 1$ and $((i \times jump) \pmod{\frac{N}{B}} = 0)$ then
        $factor = factor + 1$;
    end
    $\mathbf{V}_i = (i \times jump + factor) \pmod{\frac{N}{B}}$;
end
```

Algorithm 1: The Function to Create an Ordering among the Blocks

3.3 Implementation details

In this section, the specific values of the parameters (P, B, b), etc. for all four ciphers are presented.

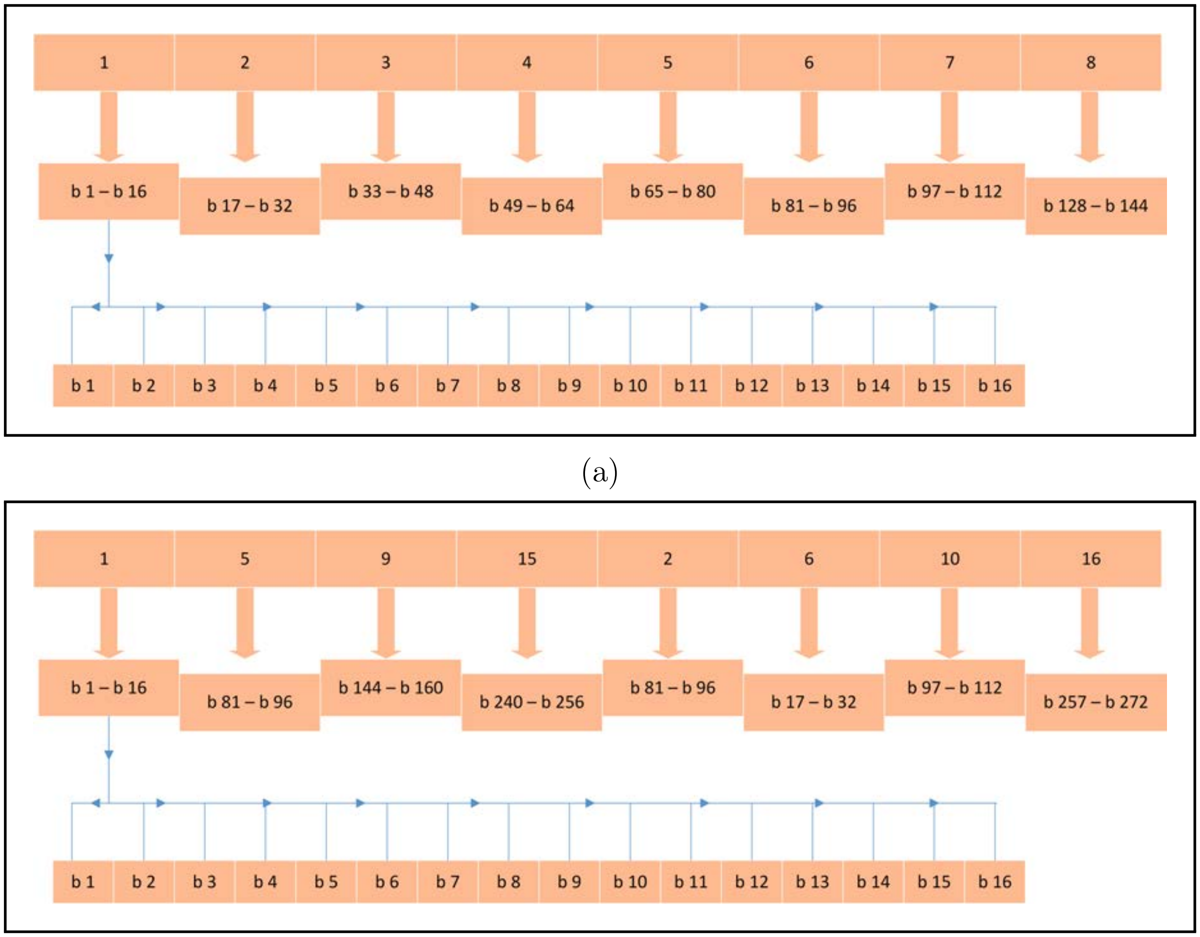
3.3.1 Specifics of AES Implementation

PARAMETER	VALUES
Algorithm	AES
Plaintext block size (b)	16 Bytes
DDR3 Memory block size (B)	256 Bytes
# banks is DDR3 (P)	8

Table 3.1: Parameters for AES Implementation

The plaintext block size b of AES is not equal to the block size B of the RAM (Table 3.1). Therefore each memory block contains $\frac{B}{b}$ plaintext blocks, which is $\frac{256}{16} = 16$ in the case of AES. Fig. 3.10 depicts the details of divisions of plaintext blocks (represented as b_i) within memory blocks (represented by integers).

Therefore as depicted in Fig. 3.10, the page table vector had to be programmed to incorporate 16 plaintext blocks per memory block to implement different ways of parallelization in case of AES.

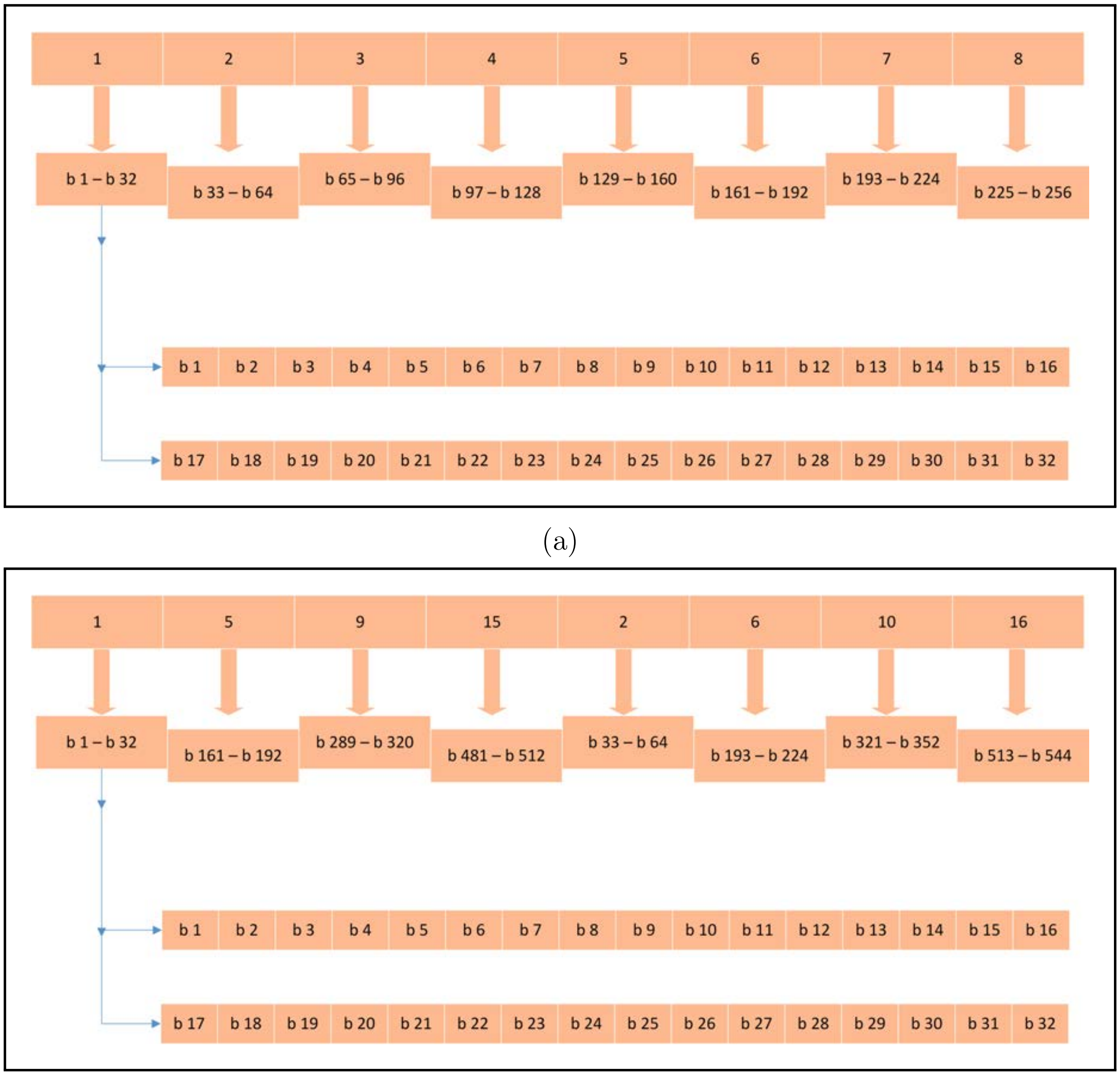


(a)

(b)

Figure 3.10: (a)AES 1-way (b) AES 4-way

3.3.2 Specifics of DES, 3-DES, and Blowfish Implementations


1
2
3
4
5
6
7
8
b 1 – b 32
b 33 – b 64
b 65 – b 96
b 97 – b 128
b 129 – b 160
b 161 – b 192
b 193 – b 224
b 225 – b 256
b 1 b 2 b 3 b 4 b 5 b 6 b 7 b 8 b 9 b 10 b 11 b 12 b 13 b 14 b 15 b 16
b 17 b 18 b 19 b 20 b 21 b 22 b 23 b 24 b 25 b 26 b 27 b 28 b 29 b 30 b 31 b 32
1
5
9
15
2
6
10
16
b 1 – b 32
b 161 – b 192
b 289 – b 320
b 481 – b 512
b 33 – b 64
b 193 – b 224
b 321 – b 352
b 513 – b 544
b 1 b 2 b 3 b 4 b 5 b 6 b 7 b 8 b 9 b 10 b 11 b 12 b 13 b 14 b 15 b 16
b 17 b 18 b 19 b 20 b 21 b 22 b 23 b 24 b 25 b 26 b 27 b 28 b 29 b 30 b 31 b 32

(a)

(b)

Figure 3.11: (a)1-way, (b) 4-way for DES, 3-DES, and Blowfish

For DES, 3-DES, and Blowfish, the plaintext block size (b) is 8 bytes (Table 3.2). Therefore each memory block contains $\frac{B}{b}$ plaintext blocks, which is $\frac{256}{8} = 32$ in the case of DES, 3-DES, and Blowfish.

PARAMETER	VALUES
Algorithms	DES, 3DES, Blowfish
Plaintext block size (b)	8 Bytes
DDR3 Memory block size (B)	256 Bytes
# banks is DDR3 (P)	8

Table 3.2: Parameters for DES, 3DES, & Blowfish Implementations

Fig. 3.11 depicts the details of division of plaintext blocks, (represented as S_i) within memory blocks (represented by integers) for DES, 3-DES, and Blowfish. Therefore as depicted in Fig. 3.11, the page table vector had to be programmed to incorporate 32 plaintext blocks per memory block to implement different ways of parallelization in case of DES, 3-DES and Blowfish.

3.4 Optimizing Symmetric Key Algorithms

As the memory layout scheme is implemented on top of the algorithm, it might lead to additional overhead, negatively impacting the energy savings. Hence some optimization techniques were incorporated by this researcher to reduce this overhead.

1. **Spatial Locality.** For every memory access a mapping function is needed, and this call creates an overhead. The concept of Spatial Locality (consecutive elements in a logical block are placed on consecutive memory locations) is used to reduce overhead.

2. **Bit shifts.** Operations like multiplication, division and mod are replaced with bit shifts wherever possible.

3. **Register variables.** To reduce runtime, register variables are utilized when a variable has to be computed many times.

CHAPTER 4

Experimentation and Results

4.1 Experimental Setup

The benchmark code was written in `C` and was compiled using the `gcc` compiler. The experiments were primarily designed to test whether change in the energy consumption of the redesigned algorithms is observed when the degree of parallelism of their inputs were varied. Hence, variations of the algorithms were implemented to allow the use of 1, 2, 4, or 8 memory banks in parallel on DDR3 memory with 8 memory banks.

4.1.1 Hardware Setup

The experiments were run on a Mac notebook with 4 Intel cores, each of which operated at a frequency of 2 GHz. The notebook had 4GB DDR3 memory with 8 banks and ran Mac OS X Lion 10.7.3. The sizes of the L2 and L3 caches per core were 256 KB and 6 MB respectively. The disk size of the machine was 499.25 GB. During any experimental run, all non-essential processes were aborted to ensure that the power measured could be attributed primarily to the application being tested. The total power drawn, including the processor power, memory (RAM) power, and background (leakage) power, was measured using Hardware Monitor tool [35]. The Hardware Monitor tool reads sensor data directly from Apple's System Management Controller, which is an auxiliary processor independent of the main computer. Hence, energy consumed by the monitoring infrastructure does not add to the sensor readings. The energy consumption for each algorithm was calculated by multiplying the power recorded with the time taken to execute the algorithm.

4.2 Results

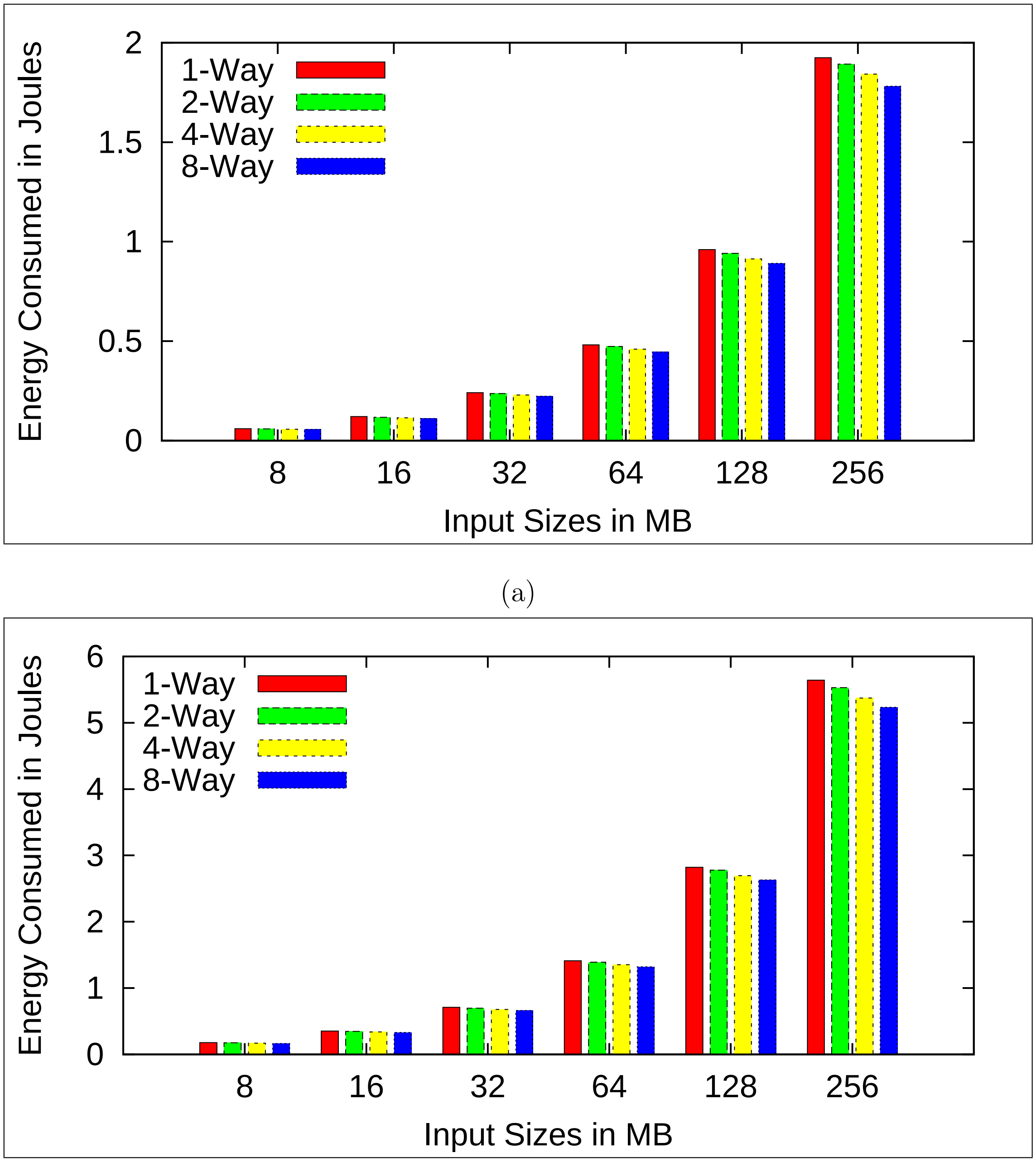
Each experiment was repeated 50 times and the means are reported here. The variation in the energy consumption, namely the deviation from the average values, was small (less than 4%) for all the experiments performed.

Algorithm	Percentage difference in Energy Consumption
DES	7.46%
3-DES	6.41%
AES	7.45%
Blowfish	6.87%

Table 4.1: 1-way vs. 8-way parallel for 16 MB plaintext input.

4.2.1 Applicability of the Energy Model

The energy savings achieved in energy consumption for the input size of 16 MB by increasing the parallelism from 1-way to 8-way is 7.46% , 6.41%, 7.45%, and 6.87% for DES, 3DES, AES, and Blowfish respectively (Table 4.1). These experiments establish that energy consumption decreases with increased parallelization of their block accesses. The results establish the applicability of the energy model of [29] on the symmetric key algorithms considered in this work.



(a)

(b)

Figure 4.1: Energy Consumption by (a) DES and (b) 3-DES

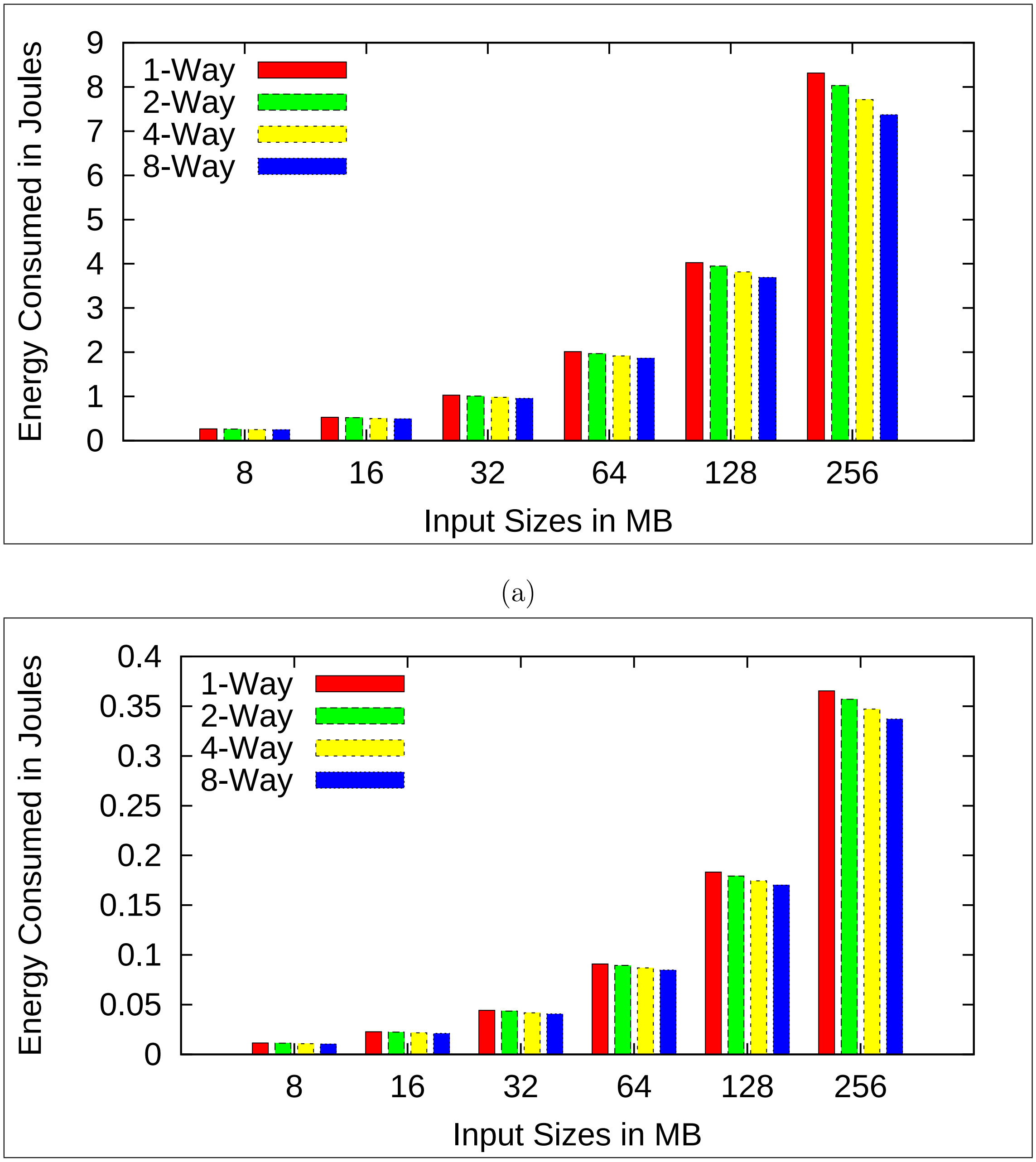
4.2.2 Comparisons based on plaintext input size

Algorithm	16 MB Input	256 MB Input
DES	7.45%	7.94%
3-DES	6.41%	8.26%
AES	7.45%	11.35%
Blowfish	6.87%	7.78%

Table 4.2: Energy savings for different input sizes.

The energy savings achieved in energy consumption for the input size of 256 MB by increasing the parallelism from 1-way to 8-way is 7.94% , 8.26%, 11.35%, and 7.78% for DES, 3DES, AES, and Blowfish respectively. Therefore, it can be concluded from Table 4.2, that the impact of parallelism is more when the input data is larger. This can be explained by the fact that smaller inputs fit into the system caches (e.g. L2, L3). The memory controller then applies smart techniques to perform write backs to make accesses more efficient (especially during low parallelism), and in the process negates the effect of parallelism for smaller inputs.

Fig. 4.1 and Fig. 4.2 depicts the results obtained from 1, 2, 4 and 8-way parallelism for all the algorithms over different input sizes. As is also depicted in the graphs, an input size of 8 MB almost does not show any change in energy consumption for different degrees of parallelization. This can be supported again by the fact the total system cache size is almost 8 MB, which exactly fits the plaintext input. For the larger inputs, the memory controller is forced to use the memory banks instead of the system caches to fit in the input. Therefore higher impact in change in energy consumption is observed for larger inputs.



(a)

(b)

Figure 4.2: Energy Consumption by (a) AES and (b) Blowfish

4.2.3 Comparison of energy consumed by different algorithms

The energy consumed by various symmetric key algorithms was also compared. Fig. 4.3 shows the energy consumed by all the four symmetric key algorithms considered in 8-way parallelism for a fixed input size of 256 MB.

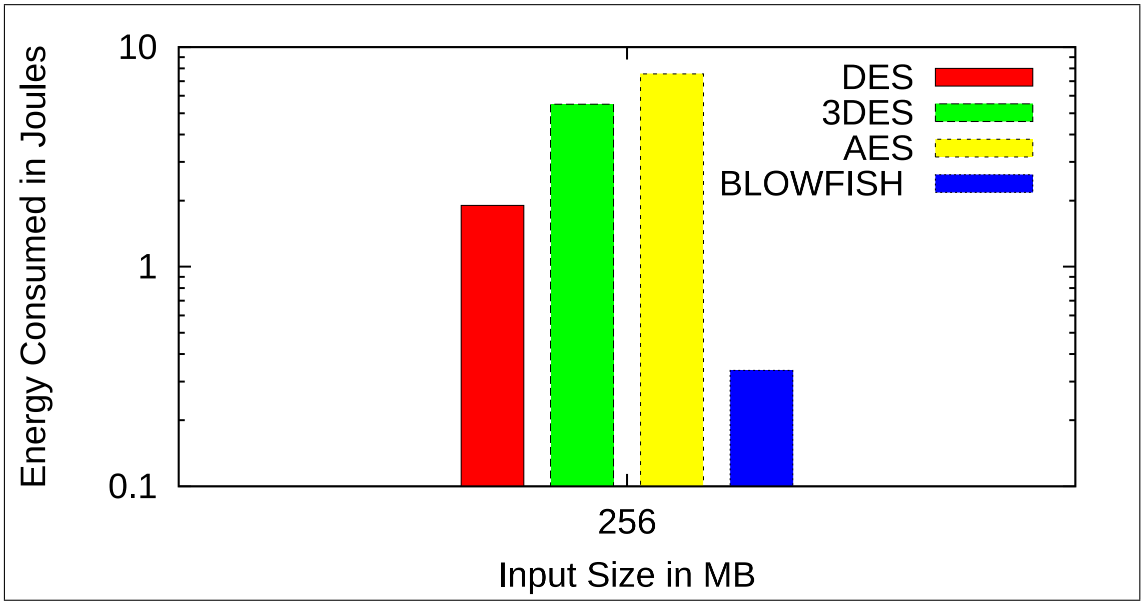


Figure 4.3: Energy Consumption by Symmetric Key Protocols

As can be seen from Fig. 4.3, AES consumes more energy than 3-DES because AES uses a 256 bit key as compared to a 168 bit key for 3-DES. 3-DES consumes about three times the amount of energy that DES consumes, which is expected. The interesting observation is how less energy Blowfish consumes.

CHAPTER 5

Conclusions and Future Work

In this chapter the results obtained in this research have been summarized, and initial ideas on further research directions in this area have been discussed. This chapter begins with the summary of results followed by discussions on future research directions.

5.1 Summary of Results

5.1.1 Applicability of the Energy Model on Symmetric Key Algorithms

In summary, the results show decreases in energy consumption with increased parallelism in all four symmetric key encryption algorithms considered, DES, 3-DES, AES, and Blowfish. This proves the applicability of the energy complexity model of [27] on a class of symmetric key algorithms. This is a significant result since this would result in the design of symmetric key algorithms that consume significantly less energy.

5.1.2 Energy Efficiency of Blowfish Encryption over other Symmetric Key algorithms

Our experiments (Fig. 4.3(b)) indicate Blowfish encryption algorithm consumes significantly less energy than its counterparts, DES, 3-DES, and AES. AES is currently the most widely used symmetric key encryption algorithm. However, there are no known attacks on Blowfish [4]. In other words, Blowfish is also a strong cipher with no known vulnerabilities. In fact, Blowfish has been found to be superior in terms of performance [24, 34] and memory usage [24] over the other three ciphers. The results in this thesis for experiments conducted to test the energy consumption of only the encryption process of the ciphers also tally with [24, 34].

To recall, there are three phases in symmetric key encryption: Key generation, Encryption, and Decryption. Key generation for Blowfish has been known to not to be computationally efficient. It is unknown how expensive the key generation algorithm is for Blowfish in terms of energy consumption. Therefore based on the results obtained here, using Blowfish can be recommended over the other ciphers, if the user is not required to change (generate) keys too often.

5.1.3 Designing Energy-Efficient AES

Additional experiments were conducted to compare the energy consumption of AES during the best case of parallelism, (8-way) with original implementation without application of the energy model (Fig. 5.1).

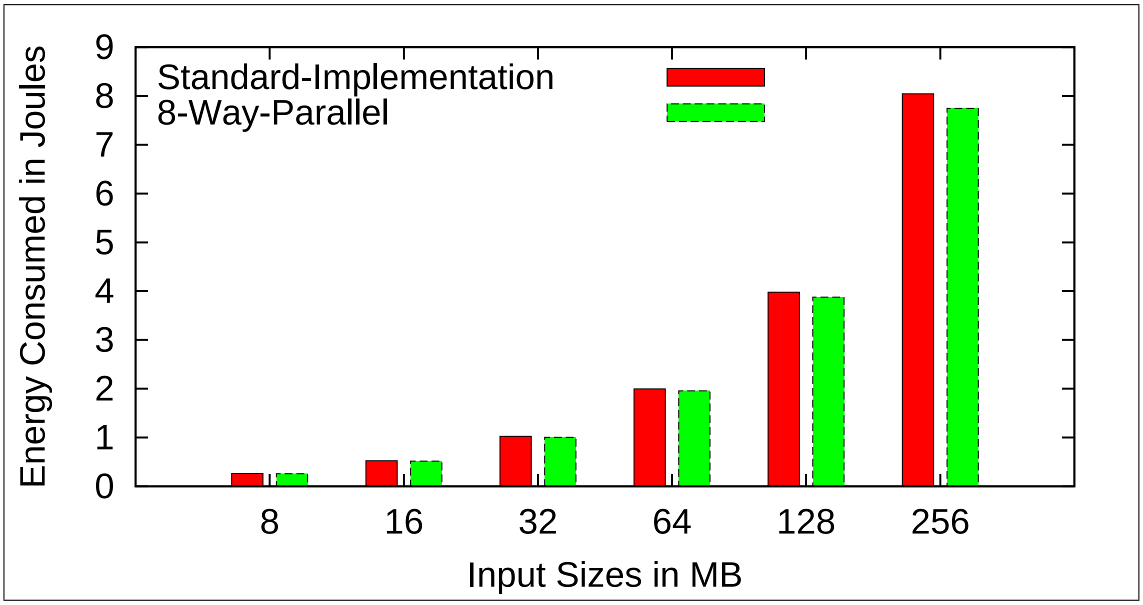


Figure 5.1: Energy Consumption by AES

The percentage decrease in energy consumption from original implementation of AES to the redesigned AES algorithm with 8-way parallelism is 3.71%. This difference is not very significant. However it can be conjectured that the difference will become large enough to be practically useful with larger inputs. Hence, it can be concluded that with large data, the 8-way parallel technique in the redesigned AES algorithm is more energy efficient than the original implementation. As a future work, similar experiments can be conducted on DES, 3-DES, and Blowfish to see how energy efficient the redesigned algorithms in 8-way parallelism are compared to their respective original implementations.

5.2 Future research directions

In this section, a few directions of further exploration are discussed.

5.2.1 Applying the Energy Model on Decryption and Key Generation

This research focuses on applying techniques for reducing the energy consumption of encryption process in symmetric key block ciphers. As a future work, the energy consumption of the decryption and key generation processes of these ciphers should be tested. It would be expected that energy consumption of the decryption process should be similar to that of encryption because decryption is the inverse of encryption in most of these ciphers. The energy consumption of key generation for DES, 3-DES, and AES should be much less compared to encryption and decryption. However, the energy consumption of key generation for Blowfish is likely to be high for the reasons described previously in this chapter, and it is important to know how high it would be in order to make decisions about using Blowfish over the other ciphers, or how could the key generation process of Blowfish be modified to make it energy efficient.

5.2.2 Applying the Energy Model on Stream Ciphers and PKI

Another possible direction of research is the application of the energy complexity model [29] on symmetric key stream ciphers and public key ciphers (PKI). Symmetric key stream ciphers are encryption algorithms that encrypt one bit at a time, in contrast to the block ciphers considered here, which encrypt one block (multiple bits) at a time. It will be interesting to see if this property of stream ciphers help implement the energy model on them making reduction energy consumption easier.

Public key ciphers exploit the hardness of one-way mathematical functions (e.g.

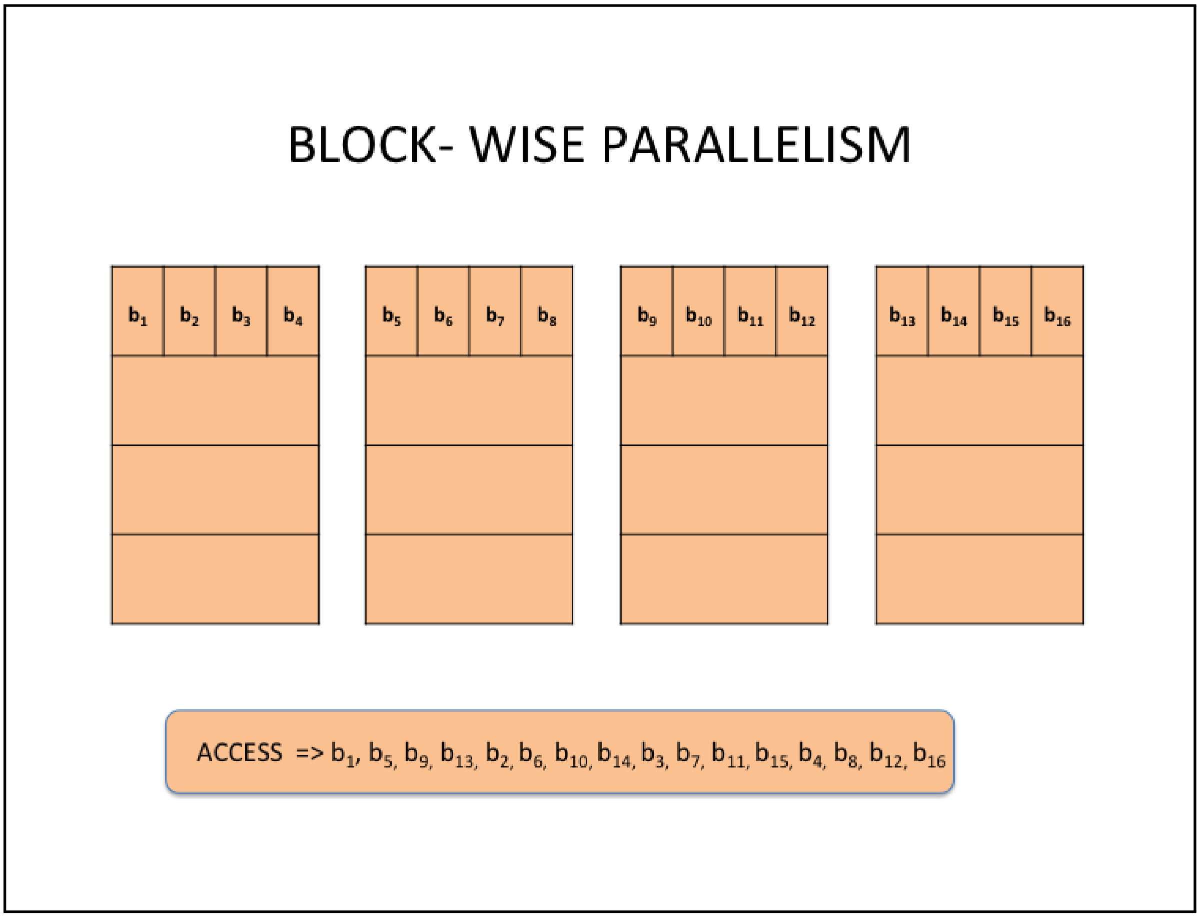
modular exponentiation, elliptic curves) for their security, in contrast to most symmetric key algorithms that uses mostly easy operations (e.g. substitution, transpositions, etc.) smartly to ensure security. In [30], the researchers implemented the energy model on a few standard mathematical functions (e.g. matrix transposition) showing reduction in energy consumption in them. The energy model might be applied to functions of public key ciphers in a similar manner.

5.2.3 Techniques to Further Reduce Energy Consumption

The maximum energy savings achieved with the redesigned AES algorithm was 3.71% compared to the original implementation (Fig. 5.1). Further research can be conducted to improve energy savings using more energy efficient techniques to parallelize block access or by using a different memory technique. A few such ideas are described in the following subsections.

5.2.3.1 Plaintext block (b)-level Parallelism

In the current work, parallelism is implemented memory block-wise (B). Energy consumption can be tested by parallelizing the plaintext blocks (b) instead. As can be observed in the Fig 5.2, the contiguous accesses in plaintext blocks that belong to different banks, can be implemented with the access pattern of $\{b_1, b_5, b_9, b_{13}, b_2, b_6, b_{10}, b_{14}, b_3, b_7, b_{11}, b_{15}, b_4, b_8, b_{12}, b_{16}\}$. This will increase parallelism over the plaintext block. This might increase energy savings in the algorithms, since the input plaintext is encrypted and decrypted according to the blocks of the plaintext (b) (and not the blocks of the memory (B)).


BLOCK- WISE PARALLELISM

ACCESS => b_1, b_5, b_9, b_{13}, b_2, b_6, b_{10}, b_{14}, b_3, b_7, b_{11}, b_{15}, b_4, b_8, b_{12}, b_{16}

Figure 5.2: Block-Wise(b) Parallelism

5.2.3.2 Multi-Threading

As described in section 3.1, a block(B) of a particular bank in the memory can be accessed only when it is in the cache of that bank. Each time a block(B) is transferred to the cache, a constant amount of power is consumed. In the current work, all the plaintext blocks(b) in a block(B) are accessed contiguously and moved on to the block(B) in the next bank, thereby saving power. If we can implement multi-threading in addition to the current work, more power can be saved. For example, consider blocks B_1 - B_{16} distributed into 4 banks as shown in the Fig 5.3.

The input is divided into 4 (total number of banks) chunks i_1 - i_4 such that each i has all the blocks belonging to the same bank. Each i is executed in a separate thread T as shown in the Fig 5.3.

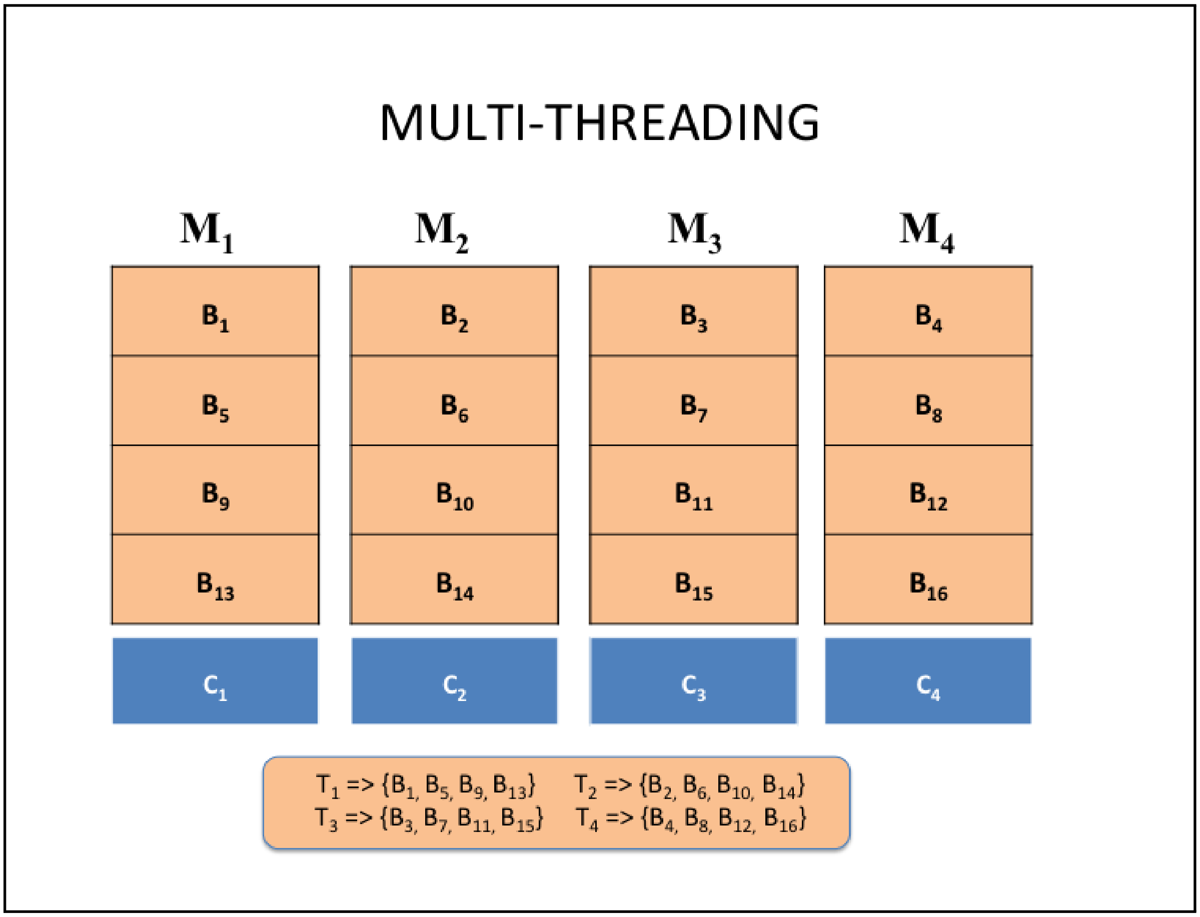


Figure 5.3: Multi-Threading

5.2.3.3 Stream Ciphers

The memory controller works in such a way that only 32-bits (in a 32-bit operating system) can be accessed in one cycle. The block ciphers can only be encrypted in blocks and the input cannot be divided any way we want. Whereas stream ciphers encrypt bit-by-bit, hence we can divide the input into 32-bit chunks. For example the input is divided into blocks of 32-bits say b_1 - b_8 and distributed into two banks M_1 and M_2 as shown in the Fig. 5.4. The consecutive access would lie in different banks i.e $b_1, b_5, b_2, b_6, b_3, b_7, b_4, b_8$, which might save energy.

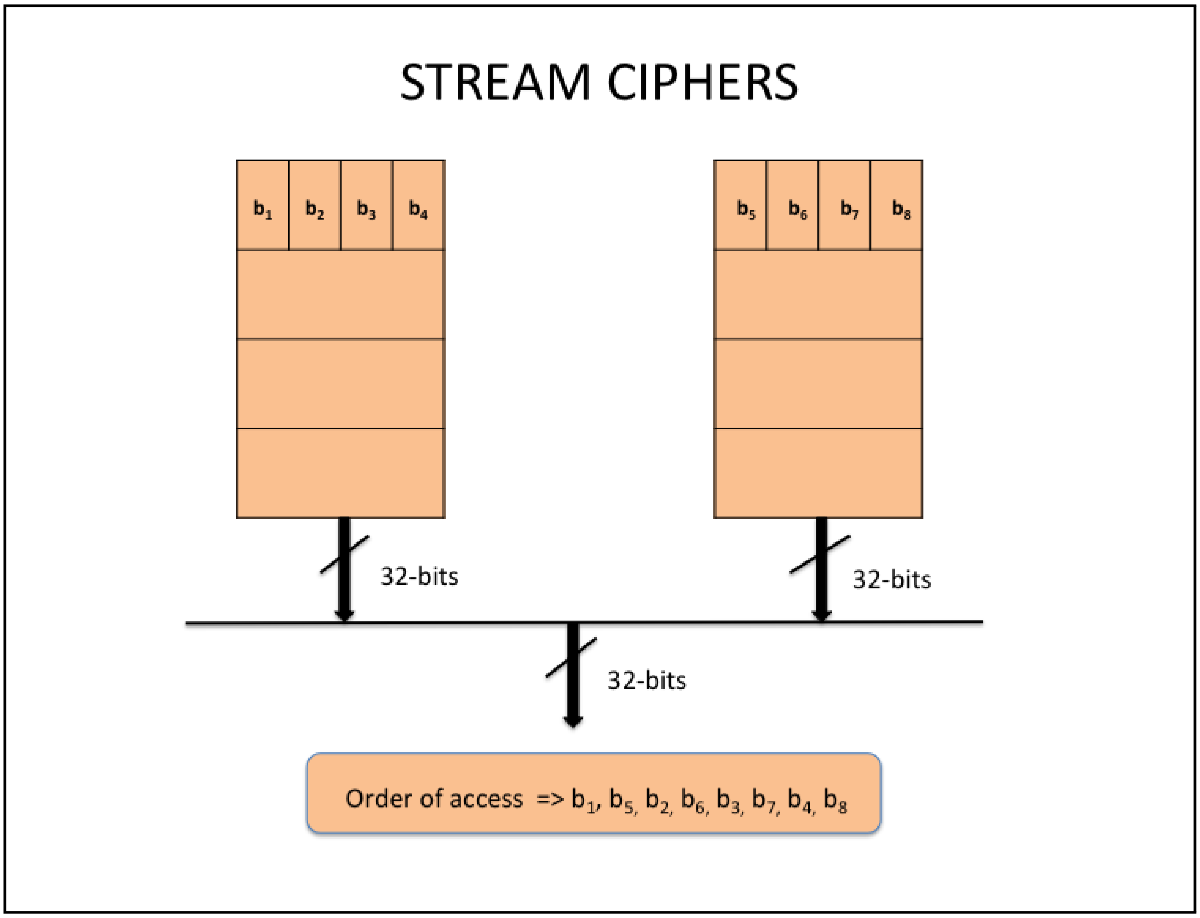


Figure 5.4: Stream Ciphers

5.2.3.4 Code Vicinity

AES comprises of constants such as round keys and S-boxes, which are used repeatedly and extensively when encrypting one plaintext block(b) in a block(B). If these constants happen to be in a different block(B) as the one currently being accessed, the current block(B) has to be put back in its original location and fetch the new block(B) into the cache. An ACT command transfers the block B to the cache C, and an ACT command is always followed by a PRE command which puts back the block B in its original location. Lets say the ACT command consumes power p_1 and PRE consumes power p_2. Each replacement would cost two times power $p_1 + p_2$, i.e. a $p_1 + p_2$ to fetch the block being accessed and another $p_1 + p_2$ to fetch the new block to cache. These two constants are used numerous times during the encryption of a single block(b).

Further work can be done to minimize these accesses. One technique would be to add the constants to every block(B). Hence with one fetch, the block(b) and, required constants can be accessed, thereby saving a lot of power. To programatically achieve this, constants can be added for every K bytes where K will be equal to the row size minus size of all the constants.

Energy savings in encryption algorithms will be increasingly important as needs for both security and portability of devices expand. This work provides a starting point for the redesign of encryption algorithms to achieve this goal.

References

[1] Aqueel, S., and Khare, K. A high performance ddr3 sdram controller. IJEEE), Volume1, Issue1 (2011).

[2] Bhutada, A., and Magar, S. Executing des algorithm in cloud for data protection. International Journal of Innovative Research in Engineering and Technology 1, 1 (2015), 15–19.

[3] Carman, D. W., Kruus, P. S., and Matt, B. J. Constraints and approaches for distributed sensor network security (final). DARPA Project report,(Cryptographic Technologies Group, Trusted Information System, NAI Labs) 1, 1 (2000).

[4] Cornwell, J. W., and Columbus, G. Blowfish survey. Department of Computer Science. Columbus: GA Columbus State University (2012), 1–6.

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

[6] Dierks, T., and Rescorla, E. The transport layer security (tls) protocol version 1.2. Tech. rep., 2008.

[7] Dong, L., Wu, N., and Zhang, X. Low power state machine design for AES encryption coprocessor. Proceedings of the International MultiConference of Engineers and Computer Scientists 2 (2015).

[8] EMVCo, L. Emv integrated circuit card specifications for payment systems. hp, 2011.

[9] Escapa, D. Encryption for password protected sections. `https://technet.microsoft.com/en-us/library/cc179125(v=office.15)`. Accessed: 2017-03-17.

[10] Falaki, H., Lymberopoulos, D., Mahajan, R., Kandula, S., and Estrin, D. A first look at traffic on smartphones. In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (2010), ACM, pp. 281–287.

[11] Farrell, T. Core architecture doubles mem data rate. Electronic Engineering Times Asia 16 (2005).

[12] Feldhofer, M., Dominikus, S., and Wolkerstorfer, J. Strong authentication for RFID systems using the AES algorithm. Cryptographic Hardware and Embedded Systems-CHES 2004. Springer Berlin Heidelberg, 2004.

[13] FIPS, P. 46-3: Federal information processing standards publication. DATA ENCRYPTION STANDARD (DES), Reaffirmed (1997).

[14] Garg, P., and Sharma, V. An efficient and secure data storage in Mobile Cloud Computing through RSA and Hash function. Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014.

[15] Garrett, K., Talluri, S. R., and Roy, S. On vulnerability analysis of several password authentication protocols. Innovations in Systems and Software Engineering 11, 3 (2015), 167–176.

[16] Gautam, A., and Jain, P. Fpga implementation of dynamic key generation to enhance des algorithm securities. International Journal of Engineering Research and Technology. Vol. 4. No 4, 01 (2015).

[17] Handy, M., Haase, M., and Timmermann, D. Low energy adaptive clustering hierarchy with deterministic cluster-head selection. In Mobile and Wireless Communications Network, 2002. 4th International Workshop on (2002), IEEE, pp. 368–372.

[18] Heinzelman, W. R., Chandrakasan, A., and Balakrishnan, H. Energy-efficient communication protocol for wireless microsensor networks. In System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on (2000), IEEE, pp. 10–pp.

[19] Hodjat, A., and Verbauwhede, I. The energy cost of secrets in ad-hoc networks (short paper). In Proc. IEEE Circuits and Systems Workshop (CAS) (2002), Citeseer.

[20] Jain, R., Madan, S., and Garg, B. Implementation and comparison of RSA and triple DES algorithm for encryption and decryption in cloud environment. International Journal Of Applied Engineering Research 10. 5 5, 19 (jul 2015).

[21] Karri, R., and Mishra, P. Minimizing energy consumption of secure wireless session with qos constraints. In Communications, 2002. ICC 2002. IEEE International Conference on (2002), vol. 4, IEEE, pp. 2053–2057.

[22] Lindsey, S., and Raghavendra, C. S. Pegasis: Power-efficient gathering in sensor information systems. In Aerospace conference proceedings, 2002. IEEE (2002), vol. 3, IEEE, pp. 3–1125.

[23] Mohammad, O. K. J., et al. Innovative Method for enhancing Key generation and management in the AES-algorithm. arXiv preprint arXiv:1504 (2015), 03406.

[24] Patil, P., Narayankar, P., Narayan, D., and Meena, S. A comprehensive evaluation of cryptographic algorithms: Des, 3des, aes, rsa and blowfish. Procedia Computer Science 78 (2016), 617–624.

[25] Potlapally, N. R., Ravi, S., Raghunathan, A., and Jha, N. K. Analyzing the energy consumption of security protocols. Proceedings of the 2003 International Symposium on Low Power Electronics and Design, 2003. ISLPED '03 (2003), 30.

[26] Prasithsangaree, P., and Krishnamurthy, P. On a framework for energy-efficient security protocols in wireless networks. Computer Communications 27, 17 (2004), 1716–1729.

[27] Roy, S. An Energy Complexity Model for Algorithms. PhD thesis, State University of New York at Buffalo, 2013.

[28] Roy, S., Ahuja, S. P., Harish, P. D., and Talluri, S. R. Energy optimization in cryptographic protocols for the cloud. In Applications of Security, Mobile, Analytic, and Cloud (SMAC) Technologies for Effective Information Processing and Management. IGI Global, 2018, pp. 24–48.

[29] Roy, S., Rudra, A., and Verma, A. An energy complexity model for algorithms. In Proceedings of the 4th conference on Innovations in Theoretical Computer Science (2013), ACM, pp. 283–304.

[30] Roy, S., Rudra, A., and Verma, A. Energy aware algorithmic engineering. In Modelling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2014 IEEE 22nd International Symposium on (2014), IEEE, pp. 321–330.

[31] Schneier, B. Description of a new variable-length key, 64-bit block cipher (blowfish). In International Workshop on Fast Software Encryption (1993), Springer, pp. 191–204.

[32] Schneier, B. Why cryptography is harder than it looks. In EDI FORUM-OAK PARK- (1997), vol. 10, THE EDI GROUP, LTD., pp. 87–90.

[33] Singh, S., and Attri, V. K. Dual Layer Security of data using LSB Image Steganography Method and AES Encryption Algorithm. International Journal of Signal Processing, Image Processing & Pattern Recognition 8 5 (2015).

[34] Singh, S. P., and Maini, R. Comparison of data encryption algorithms. International Journal of Computer Science and Communication 2, 1 (2011), 125–127.

[35] Software-Systeme, M. B. Hardware monitor. `http://www.bresink.com/osx/HardwareMonitor.html`. Accessed: 2015-07-01.

[36] Talluri, S., and Roy, S. Cryptanalysis and security enhancement of two advanced authentication protocols. In Advanced Computing, Networking and Informatics- Volume 2, M. Kumar Kundu, D. P. Mohapatra, A. Konar, and A. Chakraborty, Eds., vol. 28 of Smart Innovation, Systems and Technologies. Springer International Publishing, 2014, pp. 307–316.

[37] Talluri, S. R., and Roy, S. Cryptanalysis and security enhancement of two advanced authentication protocols. In Advanced Computing, Networking and Informatics-Volume 2. Springer, 2014, pp. 307–316.

[38] Talluri, S. R., and Roy, S. Towards designing a greener advanced encryption standard (aes). In Proceedings of the International Conference on Security and Management (SAM) (2014), The Steering Committee of The

World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), p. 1.

[39] Xiao, S. W., et al. High-speed parallel implementation of AES key expansion algorithm based on FPGA. Applied Mechanics and Materials 719 (2015).

[40] Zhu, X., Yu, W., and Jin, G. An Encryption Technique for Measurement and Display of Radar Parameters. First International Conference on Information Sciences, Machinery, Materials and Energy, 2015.

LIST OF PUBLICATIONS FROM THE THESIS

1. Talluri, S. R., & Roy, S. (2014). Towards Designing a Greener Advanced Encryption Standard (AES). In Proceedings of the International Conference on Security and Management (SAM) (p. 1). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

2. Roy, S., Ahuja, S. P., Harish, P. D., & Talluri, S. R. (2018). Energy Optimization in Cryptographic Protocols for the Cloud. In Applications of Security, Mobile, Analytic, and Cloud (SMAC) Technologies for Effective Information Processing and Management (pp. 24-48). IGI Global.

VITA

Raghu Talluri was born in , India . After finishing his school in 2009, he studied Computer Science in Jawaharlal Nehru Technological University (JNTU). He is pursuing his Masters in Computer Science in University of North Florida and working as a Software Engineer at CSX Technology. He has coauthored the following publications [15, 28, 37, 38]. His paper titled “Cryptanalysis and security enhancement of two advanced authentication protocols” [37] won the Best Paper Award at ICACNI 2014.