Ministry of Higher Education and Scientific Research
University of Babylon
College of Science for Women
Department of Computer Sciences

Performance Analysis of the Speck Cryptography Algorithm

A Project

Submitted to the Council of the College of Science for Women at University of Babylon in Partial Fulfillment of the Requirements for the Degree of High Diploma in Science \ Computer Science

Submitted by

Dunay Abd AL-Hussain Yassar

Supervised by

Asst. Prof. Dr. Ahmed Badri Muslim

2022 A.D 1444 A.H

Supervisors Certification

I certify that this Project entitled "**Performance Analysis of the Speck Cryptography Algorithm**" was done by (**Dunay Abd Al- Hussain Yassar**) under my supervision.

Signature:
Name: **Asst. Prof. Dr. Ahmed Badri Muslim**
Date: / / 2022
Address: University of Babylon/College of Sciences for Women

The Head of the Departments Certification

In view of the available recommendations, I forward the project entitled "**Performance Analysis of the Speck Cryptography Algorithm**" for debate by the examination committee.

Signature:
Name:
Date: / / 2022

Address: University of Babylon/College of Sciences for Women

بِسْمِ اللَّـهِ الرَّحْمَـٰنِ الرَّحِيمِ

فَتَعَالَى اللَّهُ الْمَلِكُ الْحَقُّ ۗ وَلَا تَعْجَلْ بِالْقُرْآنِ مِن قَبْلِ أَن يُقْضَىٰ إِلَيْكَ وَحْيُهُ ۖ وَقُل رَّبِّ زِدْنِي عِلْمًا)

صدق الله العلي العظيم
سورة طه – الآية ١١٤

Dedication

To the one who encouraged me to persevere all my life,
to the most prominent man in my life
(Dear father)
To the one in whom I rise, and upon whom I rest, to the
giving heart
(My beloved mother)
To my support in this life
(Dear brother)
To my family, to my friends and colleagues...
To everyone who contributed even a letter to my
academic life.....
To all of them: I dedicate this work, which I ask God
Almighty to accept sincerely.

Dunay Abd AL- Hussain
2022

Acknowledgment

I express my sincere gratitude to my supervisor, **Dr. Ahmed Badri Muslim**, for the continuous support of my work. He always motivates me in solving my problems. His guidance helped me in all the time of research and writing of my thesis. Finally, I would like to thank my parents for their constant love, support and encouragement. Without them none of this would have been possible. I cannot begin to thank them enough, and they will always have my respect and love.

Lastly, I would also like to thank people who have helped me and inspired me during my Higher Diploma study.

Dunay Abd AL- Hussain

2022

Abstract

Encryption algorithms such as were built in particular to guarantee security, whose design clarity is vital. The multiplicity of the planned use cases, however, needs flexibility in execution. In the building of cryptosystems, simplicity, security, and flexibility are continuous but incompatible aims. The requirement for a lightweight block cipher algorithm that supports a broad range of systems, architectures, and block/key sizes efficiently has been developing recently. In 2013, the National Security Agency (NSA) recommended the block ciphers SPECK. Different block sizes, including 16, 32, and 64 bits, and key sizes, such as 64, 96, and 128 bits, are supported by SPECK. We used speck algorithm because it is encrypted in software, while Simon algorithm is used in hardware. In this research, evaluate the performance of lightweight SPECK cryptographic block ciphers utilizing multiple criteria such as execution time, throughput, and energy usage. Utilize two distinct Intel CPUs to evaluate the Speck Cryptography technique. Three keys of sizes 128, 192, and 256 are employed in this project to compare the performance criteria. The obtained results over two different Intel processors show a linear and non-linear relationship to the key size. In other words, the increase in key length increases the execution time and energy consumption, while throughput results decrease.

Keywords: Speck cipher, Cryptography, Throughput, Energy consumption

Table of Contents

List of Abbreviations

Abbreviations	**Full Meaning**
ASIC	Application Specific Integrated Circuits
IOT	*Internet of Things*
CPU	Central Processing Unit
ICT	*Information and Communications Technology*
SCADA	*Supervisory Control and Data Capital*
PLC's	Programmable Logic **Controller**
AES	Advanced Encryption Standard
KDF	*Kenya Defense Forces*
SPN	*Service Principal Name*
ARX	Add-Rotate-Xor
CBC	Cipher Block Chaining
NSA	National Security Agency
CTR	*Click-Through Rate*
ETRI	*Electronics and Telecommunications Research Institute*

List of Figures

List of Tables

Chapter One

General Introduction

Chapter One

General Introduction

1.1 Introduction

To provide high security, it is preferable to use cryptographically powerful elements and to advance an algorithm far more frequently than might initially appear appropriate. Effectiveness, a competing goal, requires us to reduce computation as much as possible [1]. The art of cryptography is how to strike a balance between these competing objectives. The reality that performance is not a clearly delineated concept adds to the difficulty. An algorithm may execute efficiently on specialized hardware (such as an ASIC), yet perform poorly on 8-bit microcontrollers [2]. Or it might permit but necessitate a lot of code, high-throughput applications on 64-bit desktop processors. Alternatively, it might be built to maximize efficiency on a certain CPU. More information and communication technology (ICT) devices are now being included into industrial automation systems due to advances in technology. Industry 4.0, often known as the fourth industrial revolution, introduces economic and creative methods while fundamentally altering established structures and technologies. These are essentially systems for industrial cyber-physical manufacturing [1]. (furthermore related to the Industrial IOT [2] more recently), where the fast development of ICT enables the creation of cutting-edge services and goods, technological enhanced tools and innovations, as well as improved production rewards while lowering expenses. Supervisory Control and Data Collecting (SCADA) systems' monitoring and data acquisition capabilities have significantly improved due to this technical advancement, but it also raises serious questions about how vulnerable they are to cyberattacks

[3],[4],[5] . The important component of SCADA systems is PLCs . Even though there virtually prominent in each sector of business, they incredibly adept at managing a variety of industrial systems. The main producers are only now beginning to pay attention to their security aspects, which were virtually missing until recently. The first step in giving these systems the necessary security may be the installation of encryption at the PLC's application level. Additionally, new advancements in the field, as in [6], [7], allow for the execution of cryptography on various devices with limited computational power. This project clarifies the implementation issues the implementation issues SPECK class of simple block cryptography in applications the of PLC. It explains that, in spite of these block ciphers' outstanding performance and potential for imposing the security requirements of the application-layer, the underlying hardware architecture must be closely scrutinized by designers, particularly the supporting data kinds. The enormous variety of SPECK variations ensures that there are enough possibilities, despite the fact that these can have a substantial influence on the efficiency of the program. A lightweight cipher is a type of encryption having a small computational complexity and/or footprint.Its objective is to increase the applications of cryptography on constrained devices, and it is now going through a global standards and guidelines compilation process. Additionally, new advancements in the field, as in [6], [7], allow for the execution of cryptography on various devices with limited computational power. Designers must carefully investigate the underlying hardware architecture despite the two block ciphers' remarkable performance and propensity to impose application-layer security constraints, the allowed data types in particular. The enormous variety of SIMON and SPECK variations ensures that there are enough possibilities, we used speck algorithm because it is encrypted in software, while Simon algorithm is

used in hardware. Despite the fact that these can have a substantial influence on the efficiency of the program.

This project, studies the performance of lightweight SPECK cryptography block ciphers using different factors such as execution time, throughput, and energy consumption.

1.2 Literature Review

SPECK in various hardware and software systems because of the potential outcomes. The efficiency of the SPECK encryption method on an MSP430 processor was examined by Buhrow et al. in [10]. Various block and key size changes were carefully investigated. [11]. Manifas et al. demonstrated the use of SPECK in integrated systems [12]. To increase the efficiency of such block cryptography, researchers in [12] show that they used an ARM - NEON system. Dual ciphers have been proven to function efficiently on 8-bit CPUs [7]. The research summary [15] includes comprehensive results and comparisons to alternative lightweight methodologies as validation of the Speck algorithm's efficiency and development benefits. The latter, on the other hand, provides no analysis of the two ciphers' cryptographic strength or security evaluation of their performance.

Moreover, predetermined public strings degrade security because they increase predictability and provide cryptanalysts with a partial understanding of the key [16]. As a result, Key derivation functions are made to produce secret keys that random. Key derivation function (KDF) is a critical random number generator used by the Key Scheduling Algorithm (KSA) to generate a cryptographic key from a user-supplied input string of arbitrary length. To prevent the ciphertext from disclosing any information about the private key, KDF must be random and cannot include known patterns [17], [18]. The work in [16] developed a novel KDF approach based on Quasigroup-based expansion that is completely key-dependent, requiring the cryptanalyst to correctly guess every element in the key string to succeed. As a consequence, the output of the algorithm is greatly affected by changes in the input.

1.3 Project Problem

Speck is a light-weight cryptography designated for low-power computing devices. Moreover, it uses different key lengths to encrypt and decrypt the plaintext. The increase in the key size will also increase the computation speed while increasing the security level.

1.4 Aim of the Project

The significance of this project is to study the performance of lightweight SPECK cryptography block ciphers which used three different lengths of the key as well as two types of processors. The three commonly used lengths of keys are (128/128, 128/192, and 128/256), using different factors such as execution time, throughput, and energy consumption. The project compare the obtained results using two types of processors, Xeon processors operating and Intel Core-i7 processors operating at 2.2 and 2.8 GHz frequencies, respectively.

1.5 Project Outline

In addition to this chapter; the project consists of three chapters as follow:

Chapter two: This chapter entitled “Theoretical Background” deals with the theoretical framework of the research such as definition and explaining the Block and Stream ciphers, the Speck cryptography algorithm is introduced and its performance in terms of transaction time, throughput, and energy usage is evaluated.

Chapter three: This chapter entitled “practical background and experimental results” ,deals with the practical framework of the research by reviewing the programs was designed and implemented to achieve best cases for energy consumption, execution time, throughput; then explore the experimental results.

Chapter four: This chapter, named "Conclusions and Suggested Future Works" explores the majority of the conclusions concerning the applicable proposed algorithms and record ideas for developing this product or case study.

Chapter five: Present of the Speak algorithm and future work through which the algorithm might be developed in this chapter entitled "Conclusions and Future Works".

Chapter Two

Theoretical Background

Chapter Two

Theoretical Background

2.1 Introduction

In the cryptography there are many factors that can affect the security level such as key size, number of rounds, and types of crypto primitives. Complex keys and crypto operation can increase the encryption algorithm execution time. However, a compromising between the security level primitives and the computation cost is required to produce fast and secure encryption algorithm. This chapter discusses the Lightweight symmetric and asymmetric key cryptography. Additionally, it explains both Block and stream cipher, and the Speck cryptography algorithm. Three different metrics, execution time and throughput and energy consumption, are presented to describe the performance of the speck cryptography.

2.2 Symmetric and Asymmetric Key Cryptography

The basis of symmetric key encryption is the exchange of a key by two parties. Figure (2-1) Symmetric key encryption. Asymmetric key cryptography uses a pair of keys that are widely available, one of which is used for encryption and the other for decryption. Figure (2-2) Asymmetric key encryption. Because it is more effective, symmetric cryptography is better suited to encrypting and decrypting enormous volumes of data. A new 'style' of encryption algorithms has evolved in the realm of cryptology: lightweight encryption, which achieves a balance among security and energy efficiency [25]. Due to the fact that "standard cryptography methods consume significant quantities of energy and bandwidth both at the source and the destination," [23], these kinds of methods seem to be the most effective in disproving the widely held belief that IoT apps cannot implement encryption efficiently.

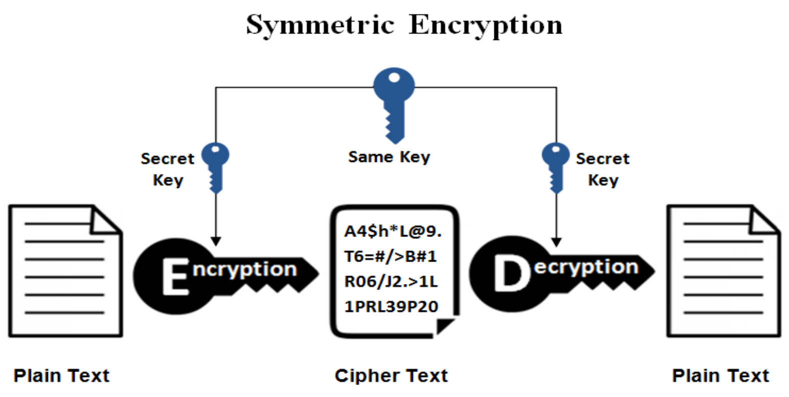

Figure (2-1): Symmetric key encryption

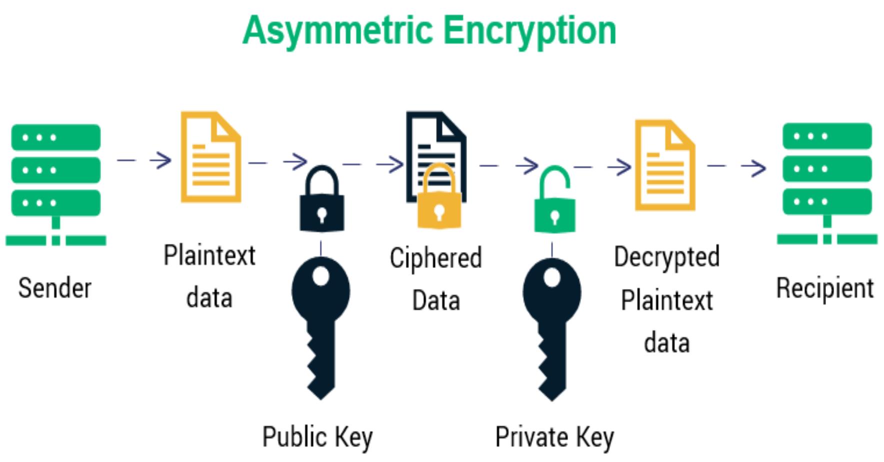

Figure (2-2): Asymmetric key encryption

2.3 Block and Stream Cipher

2.3.1 Block Ciphers

Block encryption encrypt a message by breaking it up into blocks with a set size of N. Every block goes through the same parameterized permutation, with the private key k serving as the parameter. The encrypted block is created by the rotation. In order to retrieve the plaintext, the inverse of the rotation should be performed to the cipher text using the same key**.** Block encryption are often built of numerous comparable rounds iterated over and again. There are several types of similar structures, including system with regard networks (SPN), addition-rotation-xor (ARX), and Cryptographic systems.

- **Modes of operation**: Block ciphers must be combined with security ways of working to securely encrypt communications. Another of the requirements is that by changing the key for each block, an attacker should not be able to determine that the same two blocks have been encrypted with the same key (which is a non-negligible operation). Two popular modes are Counter Mode (CTR [3]) and Cipher Block Chaining (CBC [4]). Additionally, authentication modes like the O set Encoding Mode, OCB [11] created by Krovetz and Rogaway can be used to construct encryption algorithm blocks.

- **AES** : Domain name and find better ways [2] created the AES (formerly known as Rijndael). The encryption process began with anywhere between 10 and 14 rounds, depending on the size of the key. It operates on 128-bit message blocks, which can be visualized as a 4 by 4 byte matrix. Four changes make up a round. Each byte of the state matrix in SubBytes is subjected to a single 8-bit S-box 16 times in succession (SB). The ShiftRows (SR) function rotates the state matrix's fourth row's four bytes positions to the left. Each column of the state matrix in Mix Columns receives an independent application of a linear modification described by an MDS matrix (MC). After that, a 128-bit subkey provided by key scheduling is inserted into the internal state of Add Rounds Key (AK) via an exclusive or nonexclusive operation. This figure (2-3) Block ciphers.

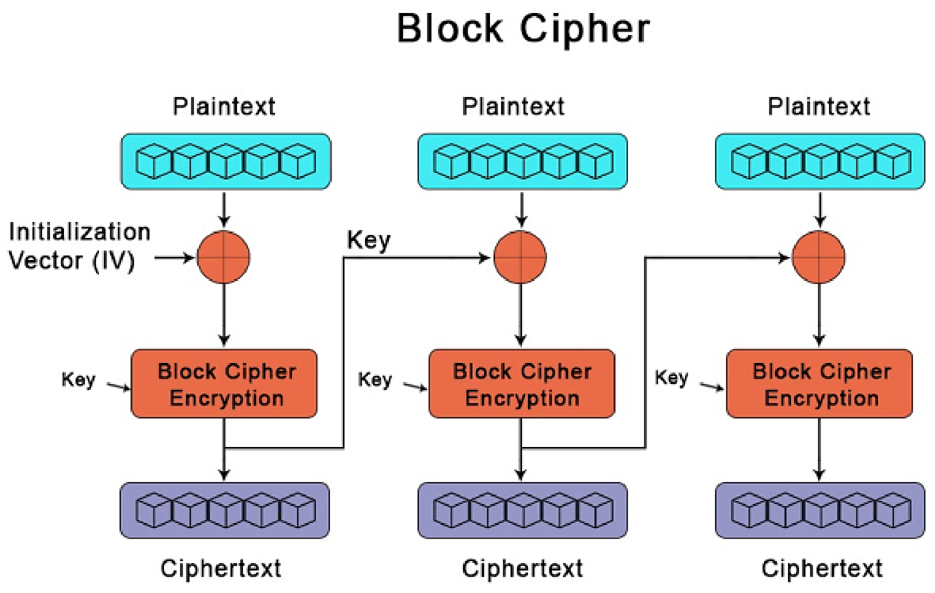

Figure (2-3): Block ciphers

2.3.2 Stream Ciphers

Stream ciphers are synchronous incremental ciphers. For instance, to create the ciphertext, each bit of the plaintext is combined by XOR with a binary hidden sequence that is the same length as the transmission (the keystream). Only one relates to the situation where the keystream is a shared, private, and distinctive string between the parties. This method guarantees complete discretion [9]. However, it is extremely impractical since need to share a key for the duration of the communication. The keystream is thus, for most practical circumstances, a pseudo-random sequence generated by a pseudo-random generator from a brief secret seed, the master key. Unless the secret key is known, the keystream must be indistinguishable from a really random distribution. To decrypt, exchanging the tiny seed enables the generation of the same pseudo-random sequence, which can then be used with the ciphertext to retrieve the plaintext. Generally the pseudo-random generator is initialized with a public value in addition to a secret key (the IV). That also enables us to alter the secret key while reinitializing the producer. This figure (2-4) Stream ciphers.

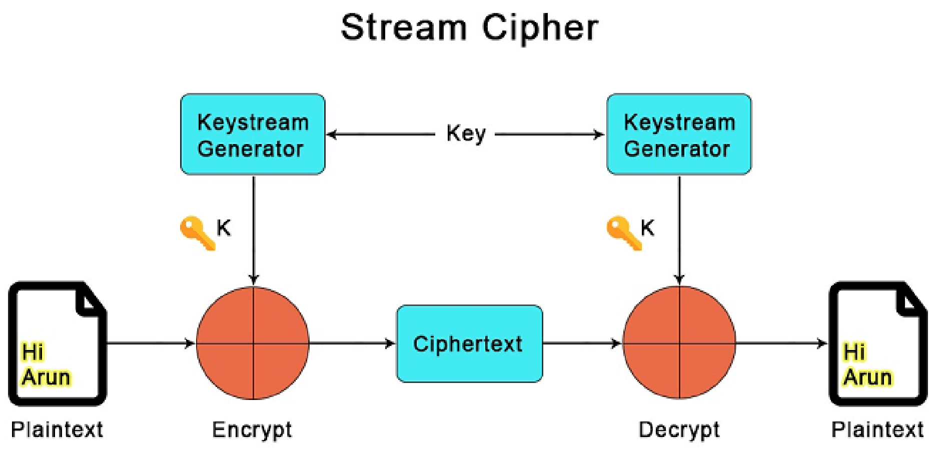

Figure (2-4): Stream ciphers

2.4 Lightweight Cryptography Algorithm

A form of encryption with a small computational complexity and/or footprint is known as lightweight cryptography. Its objective is to increase the applications of cryptography on constrained devices; it is now going through a global standards and guidelines compilation process. The characteristics of lightweight cryptography have already been explored in ISO/IEC 29192 in ISO/IEC JTC 1/SC 27. The ISO/IEC 29192 standard initiative for novel lightweight cryptography is currently being standardized. The lightweight properties of target platforms are in-depth detailed in ISO/IEC 29192. Consumption is a key metric to assess lightweight qualities in hardware implementations. Smaller code and/or RAM sizes are preferred in software implementations for lightweight applications. Lightweight cryptography provides appropriate security as well. Lightweight encryption does not always take advantage of protection trade-offs. Present new lightweight cryptographic primitives technologies.

2.4.1 Speck Cryptography

The National Security Agency (NSA) made the Speck family of simple block ciphers available to the public in June 2013 [3]. Performance optimization has been made to Speck for software applications. The Speck cipher is an add-rotate-xor (ARX) cipher. In 2011, the NSA build the Speck ciphers. They foresaw that some US federal government organizations need a cipher in variety of IoT devices and retaining a respectable level of security. The development of KDF relies on specific programming operations to be used in altering binary values of pseudo vector parameters formed the parallel implementation of algorithmic modules in order to produce key variables so the algorithm abruptness and use it for key scheduling. First, to improve SPECK one must change how it operated by adding a key derivation function. Since SPECK was created for resource-constrained environments, the researchers came up with a method that does not affect performance by using simple bitwise operations. The K-SPECK 128/256, and the K-SPECK 128/128, and the K-SPECK128/192 are three of the models that were offered. These three options are taken into consideration since, according to the Federal government, Minimum security for cryptographic protection is 112 bits. [23]. The length requirements for the plaintext and user-supplied key inputs are necessary for the SPECK CTR strategy to produce ciphertext. During scheduling the key and the encryption processes, the key derivation function algorithm turn to generate the key for the algorithm [24].

2.4.1.1 Speck Round Algorithm

For each round, SPECK performs the following three fundamental operations on an n-bit word:

– First, do a bitwise XOR, $\oplus$,

– Add by multiplying by 2n,

– Perform left and right circular shifts by r2 and r1, respectively.

To the r-th round, the left half of an n-bit word is marked by Xr1, L, the right half by Xr1, R, and the n-bit round key used in the r-th round is denoted by kr. The words Xr,L and Xr,R represent the output of round r and are calculated as follows:

$$L ,Xr =((Xr-1,L \gg r1)Xr-1,R) \oplus kr \qquad (1)$$

$$R,Xr =((Xr-1,R \ll r2) \oplus Xr,L) \qquad (2)$$

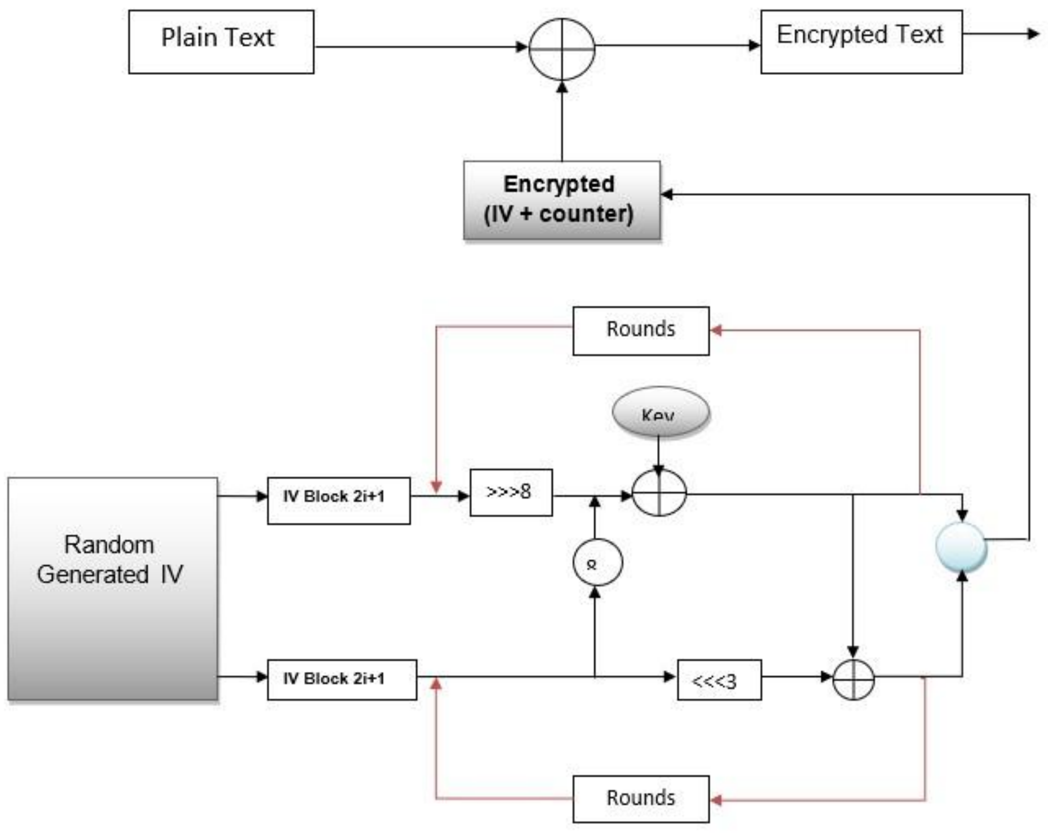

Figure (2-5): SPECK round function

Derivation of Keys Algorithm

```
  PrKey ← Pi ⊕ Pj|| P2 ⊕ Pk|| Pk ⊕ Pl|| Pl⊕ Pi’
  Q ← PrKey = 0..n
  for p = 0..a
     k ← Pa+1 >>>aŋ
  end for
```

Algorithm for Key Expansion

```
for  j = 0..T-1
  ℓ [j+n-1] ← (k[j] + S−α ℓ[j]) ⊕ j
  k[j+1] ← Sβ k[j] ⊕ ℓ[j+n-1[
end for
```

Next, given kj−m−1,…,kj−2kj−m−1,…,kj−2 . Generate the master key by repeatedly inverting the key schedule.

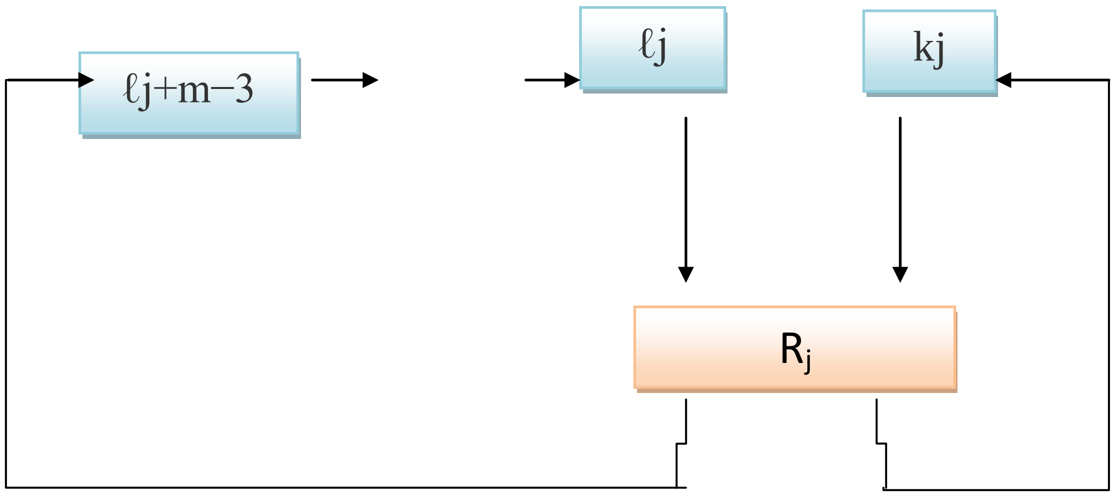

Figure (2-6): SPECK key expansion.

2.2.1.2 Speck Round Function

```
void spck_rnd (uint64_t *h, uint64_t *g, const uint64_t *k)
{
*x = (*h >> 8) | (*x << (8 * sizeof(*h) - 8));     // Rotate Right
*h += *g;
*h ^= 2;
*g = (*g << 3) | (*g >> (8 * sizeof(*g) - 3));     // Rotate Left
*g ^= *h;
}
```

***Where*:**

n = size of word (64,48, 32)

m = how many crucial words (must be 6 or 5

if n = 3,2, or 32

if n = 5, 2 ,42 or 5 or 6

if n = 64)

T = amount of rounds = (26 or 27

if n = 32

m = 3 or 4 28 or 29

if n = 48

m = 3 or 4, 32, 33or 34

if n = 3 or 4

m=64)

α, β, ŋ = (8, 3, 7)

x , y = plaintext

r, p = tempkeygen

ℓ[m-2] .. ℓ[0] , k [0] = key words

Q = rlut

2.2.1.3 Speck Encryption CTR 128

```
void spck_encrpt ( uint64_t key [2], uint64_t initial _v[ 2 ],
 uint64 _t ciphertext [2])
{
   ciphrtxt [0] = t initial_v [0];
```

```
    ciphrtxt [1] = t  initial_v [1];
    speck_round (&ciphertext [0], &ciphrtxt [1], &key [0]);
     for  (int  i  = 0;  i  < rounds;  i++ )
     {
         speck_round (&key [0], &key [1], i);      // counter mode
        speck_round (&ciphertext [1], &ciphertext [0], &key [2]);
     }
}
```

2.4.2 Simon Cryptography

Simon is a lightweight block cipher family that was publicly revealed during June 2013 by the NSA (National Security Agency), [5][1]. While its sibling approach, Speck, has been optimized for software implementations, Simon tailored for performance in the implementation of the hardware. It started working on the Simon cipher and Speck cipher during 2011. According to organization, numerous agencies of US federal government would need a cipher that would function well on many devices of Internet of Things and still maintain a respectable level of security. The 10 Speck instances were created with outstanding hardware and software performance in mind, with a particular emphasis on microcontroller performance. Utilize the same nomenclature for all of the different Speck types that Speck does. For instance, "Speck 96/144" refers to the Speck block cipher with a 96-bit block size and a 144-bit key size. Simon 2n designates the Simon block cipher with a word of length n (and hence a block of length 2n), where n must be one of the following: 16, 24, 32, 48, or 64. Simon 2n/mn is the name for Simon 2n with an m-word (mn-bit) key. For instance, Simon 64/128 designates the Simon variant that uses 128-bit keys and 64-bit plaintext blocks. The well-known Feistel law of motion is applied in every occurrence of Simon.

Without losing software speed, the approach was created to be extremely hardware compact and simple to serialize at various levels.

2.4.3 LEA Cipher

The LEA (Low-power Encryption Algorithm) developed in 2013 by the Attached Institute of ETRI [4]. It offers excellent performance in both software contexts and hardware contexts due to its straightforward Addition-Rotation Exclusive-Or (ARX) and non-S-box design. LEA is a 128-bit block cipher with 32-bit words. You can pick between a number of security settings, including 128-bit, 192-bit, and 256-bit. The rounds required for 128-bit, 192-bit, and 256-bit keys are 24, 28, and 32, respectively. Key scheduling, encryption, and decryption procedures are all included in the algorithm.

Suggest LEA, a new block cipher with a key size of 128 and 192 and 256 bits with a block size of 128 bits. It enables quick software encryption on general-purpose CPUs. This project shows the Intel, AMD, ARM, and ColdFire computers. Code size can be decreased via LEA as well. High throughput per area is achieved through its hardware implementation. It can withstand any block cipher attack pattern.

A 128-bit block cipher called the Lightweight Encryption Algorithm (LEA) was developed in 2013 by South Korea to secure privacy in both high-speed contexts like big data and cloud computing and lightweight contexts like IoT devices and mobile devices[1] . Three key lengths are supported by LEA: 128, 192, and 256 bits. In a variety of software situations, LEA encrypts data times faster than AES with 1.5–2 times. The State of Korea's national standard for cryptography is LEA, which has been approved by the Korean Cryptographic Module Validation Program (KCMVP) (KS X 3246).

2.5 Performance Evolution Metrics

Trying to mask an implement company will incur performance of the learners in terms of the number of functions conducted as well as the random generated**.** The system's performance is measured by its throughput capacity and energy consumption. Both these metrics depend on the execution time of the application. The Speck cryptography algorithm uses different key sizes, each of which needs more rounds to encrypt or decrypt the plain message. However, the number of rounds in the speck algorithm increases the required execution time to encrypt or decrypt the message. The main goal of this work is to study the performance cost of the speck algorithm while using different key lengths. The built SPECK to be block cipher method in order to enable this flexibility: This permits a maximum of three key sizes for each block size and block sizes of 32, 48, 64, 96, and 128 bits. There are eleven algorithms available in this family. The different SPECK block and key sizes are displayed in bits in Table (2-1).

Table (2-1) parameters of SPECK.

size of block	sizes of key
32	**46**
48	**72. 96**
64	**96,.128**
96	**96 .144**
128	**1281,1921,256**

flip-flops included to store the key and state, logic for carrying out data encryption and key management, an instruction set for controlling cryptography, and a logic for enabling the loading of plaintext and reading out of ciphertext in the areas made available for methods. However, the decryption method is not used in any of region statistics. This is consistent with other authors' writing: A block cipher in encrypt-only mode may be preferred for small and light applications.

2.5.1 The Execution Time

To determine the running duration, identify the most known nested loops it iterated through a large percentage of data. Nested loops used in Numerous algorithms, where the outer loop iterates over one input n and another input m. The complexity of time is O (nm). The time complexity is a measurement of how much longer an approach will require to run (in terms of number of operations) as the input size increases rather than the specified number of operations an algorithm will perform. This merely examines the proportional time of the biggest elements of the algorithm: investigate the running time in respect to an input of size n rather than the runtime for a single input. Since the loop iterates through the entire array only once in the example above, it runs in precisely the same amount of time as n, indicating a linear running time.

2.5.2 Throughput

Because Speck was specifically designed to provide high throughput independent of platform, their implementations should offer solutions to problems within the lightweight encryption design space (i.e., high throughput ASICs or FPGAs or small code size on microcontrollers, etc.).Is the speed that encryption is able to consume input and generate output. During the execution time T, the throughput of encrypting or decrypting a message of size N is determined as follows:

$$\text{Thoughpot } (GigabitPerSecond) = \frac{Message\ size\ (GigaBit)}{\text{Execution time (Sec)}} \tag{1}$$

Many scholars in the literature, including [19], employ the aforementioned equation.

2.5.3 Energy Consumption

Energy efficient algorithms use less processing power, run fewer CPU cycles, and require less memory, allowing running algorithms on smaller devices that previously could not execute machine learning on the edge. While security is definitely the most critical issue to consider when selecting algorithms and setup, it isn't the only consideration, particularly in IoT-usage. Energy consumption, or cost, is an important consideration when deciding on an algorithm.

The energy consumption of a processor executing an application in the execution time unit T can be calculated by multiplying the power consumption of a processor by the execution time [20],[21],[22]. Thus, the energy consumption is computed as follows:

$$Energy\ (Joule) = Prcessor_{power\ consuption}(W) * Execution\ time\ (Sec) \qquad (2)$$

2.6 Summary

This chapter provides details on the encryption of lightweights and introduces the speck algorithm for the encryption of messages. It also describes the three metrics of execution time, throughput, and energy consumed that is used to evaluate the cryptography algorithm.

Chapter Three

Proposed Method

Chapter Three

Proposed Method

3.1 Introduction

In this chapter, I evaluate the Speck family of block ciphers, especially cipher block speck of size 128, as a cipher component that provides independent encryption, decryption, and key scheduling capabilities, and its impact in terms of execution time, throughput, and power consuming in two processors.

3.2 Speck Encryption / Decryption Algorithm

SPECK 128/128 and SPECK 128/256 are three of models that were offered. These three options were taken into consideration since, according to the Federal government; cryptographic protection must have a minimum-security level of 112 bits [23]. The length requirements for the plaintext and user-supplied key inputs are necessary for the SPECK CTR strategy to produce ciphertext. The key derivation function algorithm serves as a key generator for the method and is utilized as key input throughout the encryption and key scheduling procedure for additional information, see [24]. This Figure (3-1) show the encryption using the SPEEK algorithm with different key lengths and its impact on the execution time, energy consumption and throughput. Notice that when the key length increases, the execution time increases, which leads to an increase in the energy consumed while the throughput decreases. We conclude from that the relationship between implementation time and energy consumed is linear and the relationship between implementation time and throughput is non-linear.

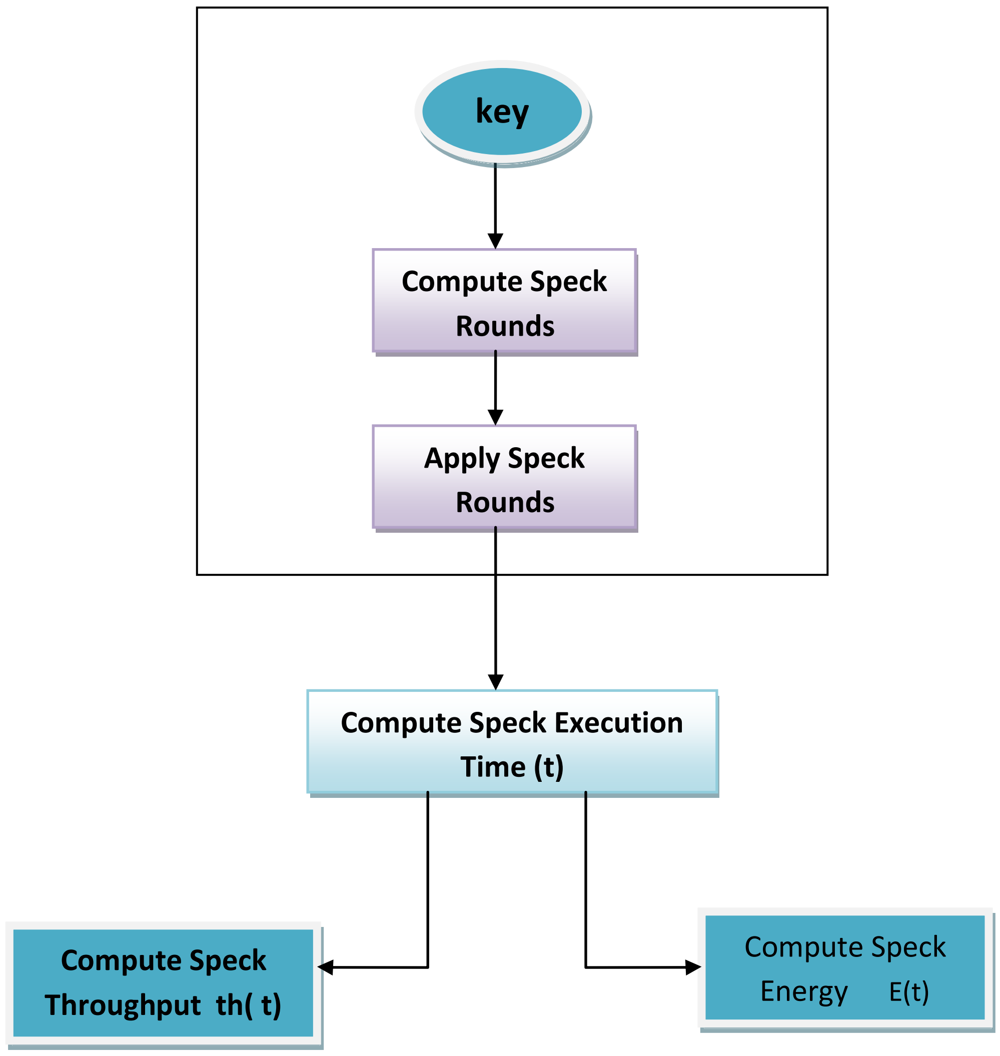
key
Compute Speck Rounds
Apply Speck Rounds
Compute Speck Execution Time (t)
Compute Speck Throughput th(t)
Compute Speck Energy E(t)

Figure (3-1): The Proposed Framework.

3.3 Summary

This chapter presents the details of the speck algorithm used and explains its structure in terms of the parts of the computation. The proposed framework for this work studies the impact of execution time, productivity, and energy consumption when running the algorithm over different generations of processors.

.

Chapter Four

Practical Framework and Experimental Result

Chapter Four

Practical Framework and Experimental Result

4.1 Introduction

This chapter displays the Speck algorithm's test results over two separate processors, Intel Core i7-700HQ and Intel Xeon. Three performance metrics are utilized, execution time, throughput, and energy consumption, in this study to explore how the key size are affects these metrics.

4.2 Experiments Setting

This part presents the outcomes of Speck cryptography, which utilized two kinds of processors and three distinct key lengths. The three commonly used lengths of keys are (128/128, 128/192, and 128/256). For each key size, there is a different number of rounds for the speck cryptography. The number of rounds is 32, 33, and 34 for the keys 128, 192, and 256, respectively. The two types of processors used in this work are Xeon processors operating and Intel Core-i7 processors operating at 2.2 and 2.8 GHz frequencies, respectively. The below Table (4.1) shows the detailed characteristics of the two used processors in the experiments. Moreover, ten message sizes were used, starting from 10 to 100 megabytes.

Table (4-1): The computing processors' technical details.

Processor Name	**Frequency speed (GHz)**	**Cache memory Size (MB)**	**Power consumption (watt)**	**Operational use**
Intel Xeon	2.2	2	65	Desktop processor
Intel Core i7-700HQ	2.8	6	45	Laptop processor

4.3 Experiments Result

The Speck cryptography algorithm is one of the lightweight cryptographies that are used in small devices. However, this section presents and analyzes its work outcomes in terms of time spent performing, energy consumption, and throughput. Two different processors were used to conduct these experiments. The goal of this project would be to explain how different software implementations affect the efficiency of the "Speck algorithm". Tables (2, 3, 4) show the obtained results of this experiments.

Table (4-2): The execution time results of the Speck algorithm.

Length of key	Size of message (MB)	The time execution (s), Intel Xeon	the Execution of Time(s) Intel ,Core, i7-700HQ
128-128	10000000	0.197562	0.198174
	20000000	0.389226	0.395368
	30000000	0.600047	0.593
	40000000	0.792915	0.790526
	50000000	0.997583	0.986128
	60000000	1.176774	1.185196
	70000000	1.392998	1.382248
	80000000	1.581127	1.568884
	90000000	1.785297	1.777749
	100000000	1.968085	1.974332
128-192	10000000	0.201467	0.203969
	20000000	0.404625	0.407419
	30000000	0.624556	0.610812
	40000000	0.82252	0.850493
	50000000	1.020373	1.01641
	60000000	1.22573	1.219265
	70000000	1.423344	1.424287
	80000000	1.631396	1.628591
	90000000	1.829164	1.831619
	100000000	2.038438	2.035171
192-256	10000000	0.221569	0.209102
	20000000	0.441812	0.4197
	30000000	0.654631	0.62788
	40000000	0.871254	0.837303
	50000000	1.095568	1.046031
	60000000	1.315916	1.265087
	70000000	1.537161	1.465924
	80000000	1.770548	1.671286
	90000000	1.974242	1.880054
	100000000	2.201328	2.087504

Table (4-3): The throughput results of the Speck algorithm.

Length of key	Size of message (MB)	Throughput (s) on Intel Xeon	Throughput Intel on Core i7-700HQ
128-128	10000000	0.424606	0.423295
	20000000	0.43104	0.424344
	30000000	0.419398	0.424382
	40000000	0.423178	0.424457
	50000000	0.420447	0.425331
	60000000	0.427709	0.424669
	70000000	0.421539	0.424817
	80000000	0.424437	0.42504
	90000000	0.422885	0.42468
	100000000	0.426232	0.424883
128-192	10000000	0.416376	0.411269
	20000000	0.414636	0.411793
	30000000	0.402939	0.412006
	40000000	0.407947	0.394529
	50000000	0.411056	0.412659
	60000000	0.410626	0.412803
	70000000	0.412551	0.412278
	80000000	0.411359	0.412067
	90000000	0.412743	0.41219
	100000000	0.411521	0.412182
192-256	10000000	0.3786	0.401173
	20000000	0.379737	0.399743
	30000000	0.384428	0.400806
	40000000	0.385128	0.400744
	50000000	0.382843	0.400973
	60000000	0.382484	0.397851
	70000000	0.382005	0.400568
	80000000	0.379029	0.40154
	90000000	0.382412	0.401571
	100000000	0.38107	0.401849

Table (4-4): The energy consumption results of the Speck algorithm.

Length of key	Size of message (MB)	Energy consumption(J) on Intel Xeon	Energy consumption(J) on Intel Core i7-700HQ
128-128	10000000	12.84153	8.91783
	20000000	25.29969	17.791559
	30000000	39.003056	26.684999
	40000000	51.539474	35.573669
	50000000	64.842896	44.375759
	60000000	76.490311	53.33382
	70000000	90.544868	62.20116
	80000000	102.773254	71.049782
	90000000	116.044304	79.998703
	100000000	127.925522	88.84494
128-192	10000000	13.095355	9.178605
	20000000	26.300625	18.333855
	30000000	40.596142	27.48654
	40000000	53.463799	38.272186
	50000000	66.324242	45.738449
	60000000	79.672447	54.866924
	70000000	92.517357	64.092918
	80000000	106.040741	73.286598
	90000000	118.89566	82.422852
	100000000	132.498474	91.582695
192-256	10000000	14.401985	9.40959
	20000000	28.717779	18.886499
	30000000	42.551014	28.254601
	40000000	56.631512	37.67835
	50000000	71.211922	47.071396
	60000000	85.534538	56.928913
	70000000	99.915466	65.966583
	80000000	115.085617	75.20787
	90000000	128.325729	84.602432
	100000000	143.086319	93.937683

The execution time results are presented in figure (4-1) for three different key lengths. As mentioned before, the key length in the Speck cipher produces more rounds when increased. However, the results of the execution time over the two processors revealed that increasing the key length increases the algorithm's execution time.

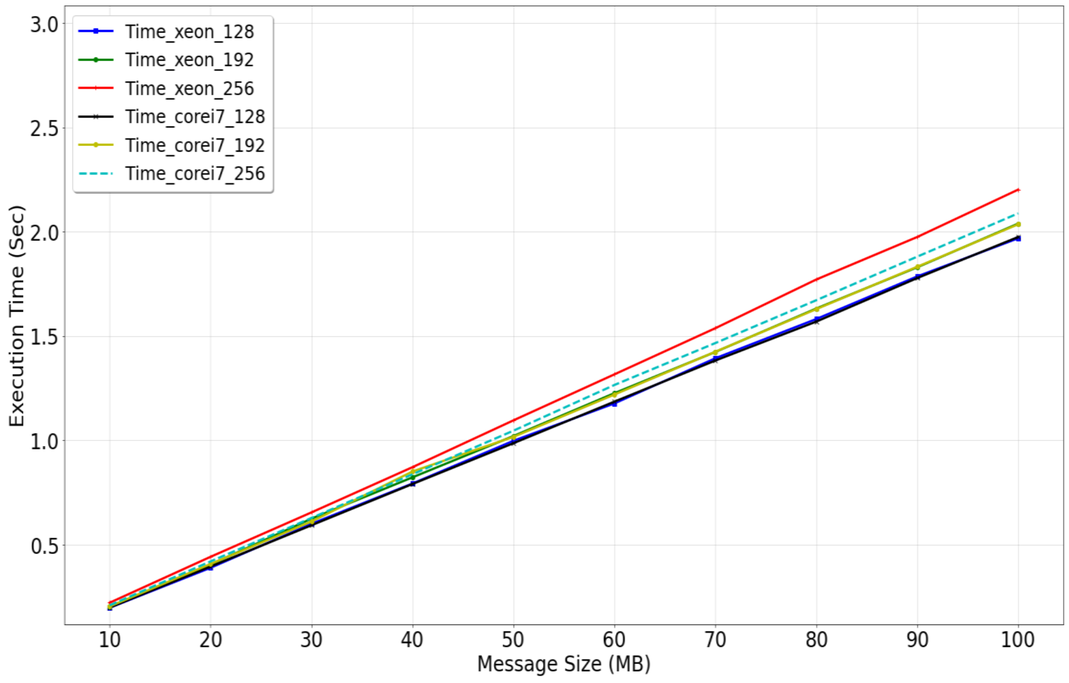

Figure (4-1): The execution time results of the Speck algorithm

Noticeably, there is a slight fluctuation that occurs in the execution time between the two processors' results. Whereas, this fact is not valid for energy consumption results when the power consumption of a processor is higher than the other one. Figure (4-2) presents the energy consumption results. It shows that the Intel Xeon processor consumed more energy with an average increase of up to 45% as compared to the Intel core i7.

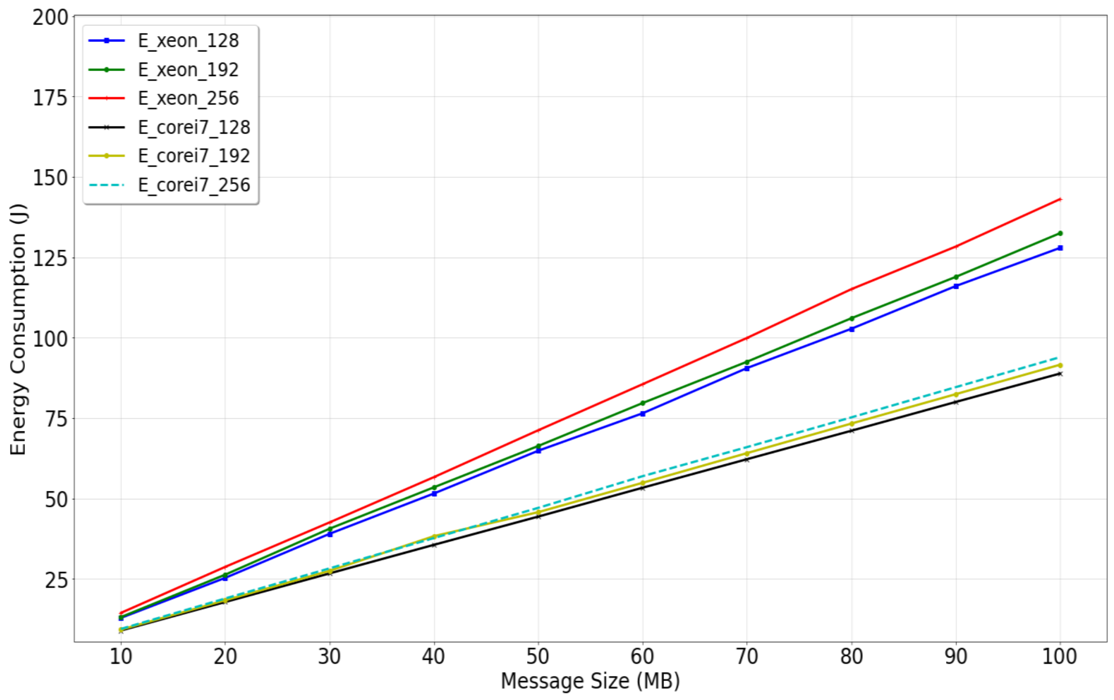

Figure (4-2): The energy consumption results of the Speck algorithm

Indeed, the energy consumption depends mainly on two factors: the processing speed and energy use of the processor. However, the Intel Xeon processor consumes more energy due to its low computing speed and high-power consumption as compared to the Intel Core i7 processor. All performance indicators can be impacted by an increase in execution time, which is significantly influenced by cache memory.

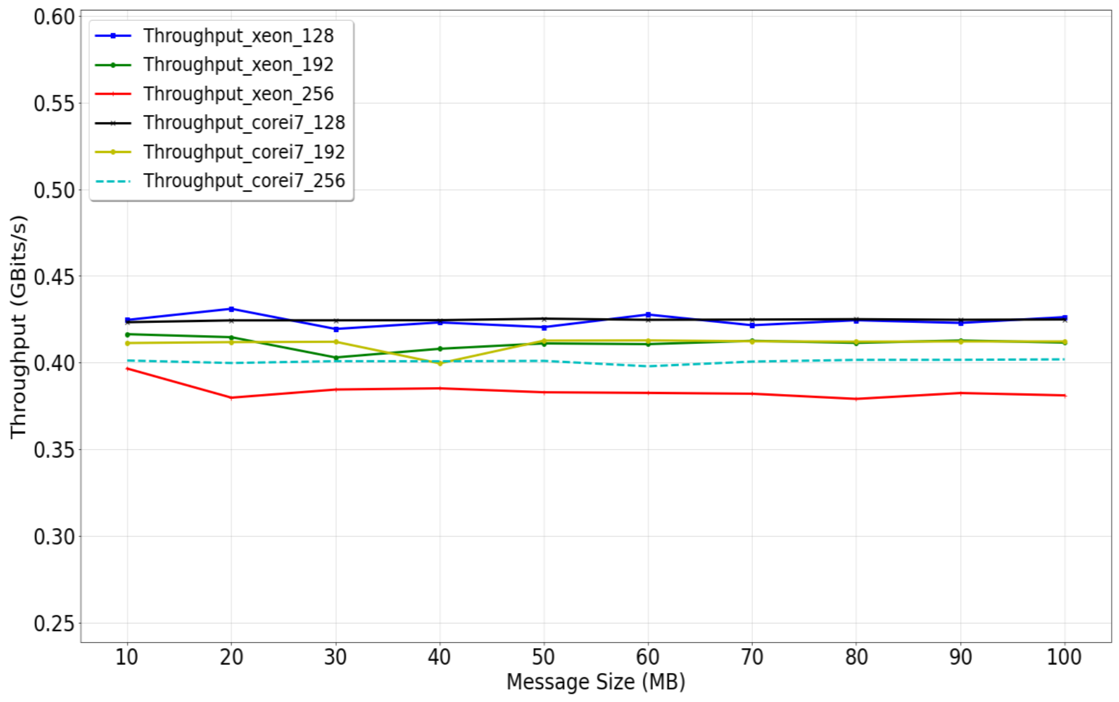

Figure (4-3): The throughput results of the Speck algorithm

The execution time, which is dependent on processor speed, determines the throughput outcomes in large part. The throughput results are presented in figure (4-3). It demonstrates that Speck with a 128-key size gives more throughputs. This is because the algorithm needs fewer rounds and thus required less execution time.

4.4 Summary

This chapter describes the obtained results of the selected approach for Speck light-weight symmetric key encryption. It displays the performance results of execution time, throughput, and energy consumption over various processors. Two different computing processors were used Intel Core i7-700HQ and Intel Xeon to conduct the Speck algorithm.

Chapter Five
Conclusions and Future Work

Chapter Five
Conclusions and Future Work

5.1 Conclusion

This project explains detailed performance study for the Speck cryptography algorithm. The algorithm used three different keys to prove how the performance metrics used change accordingly to their lengths. The performance metrics used are execution time, energy consumption, and throughput. Moreover, two Intel processors are utilized in the experiments. These processors are different in their computing speed and power consumption. The results explain that the execution of time is increased while the key length is also increased. Whereas energy results are affected by the increment of execution time and power usage. The throughput results explain the non-linear relation to the execution time.

5.2 Future Work

In the future, speck cryptography will be implemented on parallel multicore processors, and the three-performance metrics are interesting for study and analysis.

References

[1] Drath, R., & Horch, A. (2014). Industrie 4.0: Hit or hype? [industry forum]. *IEEE industrial electronics magazine*, *8*(2), 56-58.

[2] Da Xu, L., He, W., & Li, S. (2014). Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, *10*(4), 2233-2243.

[3] Hagerott, M. (2014). Stuxnet and the vital role of critical infrastructure operators and engineers. *Int. J. Crit. Infrastructure Prot.*, *7*(4), 244-246.

[4] Goodin, D. (2016). First known hacker-caused power outage signals troubling escalation. *Ars technica*, *4*.

[5] Liang, G., Weller, S. R., Zhao, J., Luo, F., & Dong, Z. Y. (2016). The 2015 ukraine blackout: Implications for false data injection attacks. *IEEE Transactions on Power Systems*, *32*(4), 3317-3318.

[6] Liang, G., Weller, S. R., Zhao, J., Luo, F., & Dong, Z. Y. (2016). The 2015 ukraine blackout: Implications for false data injection attacks. *IEEE Transactions on Power Systems*, *32*(4), 3317-3318.

[7] Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., & Wingers, L. (2015, June). The SIMON and SPECK lightweight block ciphers. In *Proceedings of the 52nd annual design automation conference* (pp. 1-6).

[8] Nithya, R., & Kumar, D. S. (2016). Where aes is for internet, simon could be for iot. *Procedia Technology*, *25*, 302-309.

[9] Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., & Wingers, L. (2015). SIMON and SPECK: Block Ciphers for the Internet of Things. *Cryptology ePrint Archive*.

[10] Buhrow, B., Riemer, P., Shea, M., Gilbert, B., & Daniel, E. (2014, September). Block cipher speed and energy efficiency records on the MSP430: System design trade-offs for 16-bit embedded applications. In *International Conference on Cryptology and Information Security in Latin America* (pp. 104-123). Springer, Cham.

[11] Manifavas, C., Hatzivasilis, G., Fysarakis, K., & Rantos, K. (2013). Lightweight cryptography for embedded systems–a comparative analysis. In *Data Privacy Management and Autonomous Spontaneous Security* (pp. 333-349). Springer, Berlin, Heidelberg.

[12] Park, T., Seo, H., & Kim, H. (2016, February). Parallel implementations of SIMON and SPECK. In *2016 international conference on platform technology and service (PlatCon)* (pp. 1-6). IEEE.

[13] Abed, F., List, E., Lucks, S., & Wenzel, J. (2013). Cryptanalysis of the speck family of block ciphers. *Cryptology ePrint Archive*.

[14] Abed, F., List, E., Lucks, S., & Wenzel, J. (2013). Differential and linear cryptanalysis of reduced-round SIMON. *Cryptology ePrint Archive*.

[15] Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., & Wingers, L. (2013). The SIMON and SPECK families of lightweight block ciphers. *cryptology eprint archive*.

[16] Disina, A. H., Jamel, S., Pindar, Z. A., & Deris, M. M. (2016, December). All-or-Nothing Key Derivation Function Based on Quasigroup String Transformation. In *2016 International Conference on Information Science and Security (ICISS)* (pp. 1-5). IEEE.

[17] Chuah, C. W., Dawson, E., & Simpson, L. (2013, July). Key derivation function: the SCKDF scheme. In *IFIP International Information Security Conference* (pp. 125-138). Springer, Berlin,

Heidelberg.

[18] Krawczyk, H. (2010, August). Cryptographic extraction and key derivation: The HKDF scheme. In *Annual Cryptology Conference* (pp. 631-648). Springer, Berlin, Heidelberg.

[19] Fanfakh, A., Noura, H. & Couturier, R. ORSCA-GPU: one round stream cipher algorithm for GPU implementation. *J Supercomput* **78**, 11744–11767 (2022).

[20] A. B. M. Fanfakh, “Predicting the Performance of MPI Applications over Different Grid Architectures”, JUBPAS, vol. 27, no. 1, pp. 468–477, Apr. 2019.

[21] Idrees, S.K., Fanfakh, A.B.M. (2018). Performance and Energy Consumption Prediction of Randomly Selected Nodes in Heterogeneous Cluster. In: Al-mamory, S., Alwan, J., Hussein, A. (eds) New Trends in Information and Communications Technology Applications. NTICT 2018. Communications in Computer and Information Science, vol 938. Springer, Cham.

[22] A. Fanfakh, J. -C. Charr, R. Couturier and A. Giersch, "CPUs Energy Consumption Reduction for Asynchronous Parallel Methods Running over Grids," *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*, 2016, pp. 205-212, doi: 10.1109/CSE-EUC-DCABES.2016.186.

[23] Barker, E., & Roginsky, A. (2018). *Transitioning the use of cryptographic algorithms and key lengths* (No. NIST Special Publication (SP) 800-131A Rev. 2 (Draft)). National Institute of Standards and Technology.

[24] Lustro, R. A. F. (2021). Modified Key Derivation Function for

Enhanced Security of Speck in Resource-Constrained Internet of Things. *International Journal of Computer Network & Information Security*, *13*(4).

جمهورية العراق

وزارة التعليم العالي والبحث العلمي

جامعة بابل

كلية العلوم للبنات قسم علوم الحاسوب

تحليل أداء خوارزمية تشفير سبيك

مشروع

مقدم الى مجلس كلية العلوم للبنات في جامعة بابل/ استيفاء جزئي لشروط الحصول على درجة الدبلوم العالي في العلوم / علوم الحاسوب

مقدم من قبل

دنيا عبد الحسين ياسر

باشرف عليها

أ .م .د .احمد بدري مسلم

٢٠٢٢ م ١٤٤٤ م

الملخص

تم بناء خوارزميات التشفير مثل تقنيات تشفير الكتلة بشكل خاص لضمان الأمن على أنظمة محدودة ، والتي يعد وضوح تصميمها أمرًا حيويًا. ومع ذلك ، فإن تعدد حالات الاستخدام المخطط لها يحتاج إلى مرونة في التنفيذ. في بناء أنظمة التشفير ، تعد البساطة والأمان والمرونة أهدافًا مستمرة ولكنها غير متوافقة. تم مؤخرًا تطوير متطلبات خوارزمية تشفير كتلة خفيفة الوزن تدعم مجموعة واسعة من الأنظمة والبنى وأحجام الكتلة / المفاتيح بكفاءة. في عام ٢٠١٣ ، أوصت وكالة الأمن القومي (NSA) باستخدام الكتلة SIMON و SPECK. يتم دعم أحجام مختلفة للكتل ، بما في ذلك ١٦ ، و ٣٢ ، و ٦٤ بت ، وأحجام مفاتيح ، مثل ٦٤ ، و ٩٦ ، و ١٢٨ بت. استخدمنا خوارمية السبيك لانها تقوم بالتشفير في البرامج بينما خوارزمية سايمون تستخدم في الاجهزة. في هذا البحث ، نقوم بتقييم أداء شفرات كتلة التشفير SPECK خفيفة الوزن باستخدام معايير متعددة مثل وقت التنفيذ والإنتاجية واستخدام الطاقة. نحن نستخدم وحدتي CPU متميزتين من Intel لتقييم تقنية Speck Cryptography. تم استخدام ثلاثة مفاتيح بأحجام ١٢٨ و ١٩٢ و ٢٥٦ في هذا المشروع لمقارنة معايير الأداء. تظهر النتائج التي تم الحصول عليها على معالجات إنتل مختلفة علاقة خطية وغير خطية مع حجم المفتاح. بمعنى آخر ، تؤدي الزيادة في طول المفتاح إلى زيادة وقت التنفيذ واستهلاك الطاقة ، بينما تنخفض نتائج الإنتاجية.

الكلمات المفتاحية: شفرات سبيك ، التشفير ، الإنتاجية ، استهلاك الطاقة