IMPLEMENTING A CRYPTOGRAPHY ALGORITHM USING RUBIK'S CUBE FOR SECURITY ON IOT DEVICES

A THESIS SUBMITTED TO
THE FACULTY OF ARCHITECTURE AND ENGINEERING
OF
EPOKA UNIVERSITY

BY

TEA PAPA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

MAY, 2024

Approval sheet of the Thesis

This is to certify that we have read this thesis entitled **"Implementing A Cryptography Algorithm Using Rubik's Cube For Security ON IOT Devices"** and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Arban Uka
Head of Department
Date: July, 25, 2024

Examining Committee Members:

Prof. Dr. Bekir Karlik (Computer Engineering) _____

Assoc.Prof. Dr. Dimitrios Karras(Computer Engineering) _____

Dr. Shkëlqim Hajrulla (Computer Engineering) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name Surname: Tea Papa

Signature: _____

ABSTRACT

IMPLEMENTING A CRYPTOGRAPHY ALGORITHM USING RUBIK'S CUBE FOR SECURITY ON IOT DEVICES

Papa, Tea
M.Sc., Department of Computer Engineering
Supervisor: Assoc.Prof.Dr. Dimitrios Karras

Because more companies and individuals were transferring data to the cloud as a result of the COVID epidemic, the significance of online communication has grown dramatically. We all depend on the net to transfer data whenever we need to send it to a different individual or organisation. It's critical that sensitive information cannot be easily captured or hijacked and used against undesired parties when sent over the internet. Businesses and individuals can use encryption to send private information over the web to ensure only the intended recipient can access it.

Mathematical equations are used to modify the data in order for any cryptography technique to encrypt and decode sensitive material. The intricacy of these equations determines how much more of the device's resources can be utilised. Because so much computational power is required, data security in the Internet of Things environment is challenging. Quantum computation is a further issue with the cryptography protocols in use today. As a result, new algorithms must be developed that take into account the limitations of IoT devices while maintaining the level of security offered by robust algorithms like AES. An IoT device will be utilised for testing an innovative cryptography algorithm that will be suggested to address this issue, and its safety and utilisation of resources will be compared to the best algorithms now in use.

Keywords: *Cryptography, Security, Networks, IoT, AES, Rubik's Cube*

ABSTRAKT

IMPLEMENTIMI I NJË ALGORITMI KRIPTOGRAFIK QË PËRDOR KUBIN E RUBIKUT PËR SIGURINË NË PAJISJET IOT

Papa, Tea

Master Shkencor, Departamenti i Inxhinierisë Kompjuterike

Udhëheqësi: Assoc.Prof.Dr. Dimitrios Karras

Për shkak të shtimit të kompanive dhe individëve që transferojnë të dhëna në "cloud" si rezultat i epidemisë COVID, rëndësia e komunikimit online është rritur dramatikisht. Ne të gjithë varemi nga rrjeti për të transferuar të dhënat kurdo që nevojiten për t'i dërguar ato tek një individ ose organizatë tjetër. Është kritike që informacioni i ndjeshëm të mos mund të kapet ose të rrëmbehet lehtësisht dhe të përdoret kundër palëve të padëshiruara kur dërgohet në internet.

Ekuacionet matematikore përdoren për të modifikuar të dhënat në mënyrë që çdo teknikë kriptografie të mund të enkriptojë dhe dekriptojë materialin e ndjeshëm. Kompleksiteti i këtyre ekuacioneve përcakton se sa shumë nga burimet e pajisjes mund të përdoren. Për shkak se kërkohet kaq shumë fuqi llogaritëse, siguria e të dhënave në mjedisin e Internetit të Gjërave (IoT) është sfiduese. Si rezultat, algoritme të reja duhet të zhvillohen që marrin parasysh kufizimet e pajisjeve IoT, duke ruajtur nivelin e sigurisë që ofrohet nga algoritme të forta si AES. Një pajisje IoT do të përdoret për të testuar një algoritëm inovativ të kriptografisë që do të sugjerohet për të adresuar këtë çështje, dhe siguria dhe përdorimi i burimeve të tij do të krahasohen me algoritmet më të mira që janë aktualisht në përdorim.

Fjalët kyçe: *Kriptografi, Siguri, Rrjete, IoT, AES, Kubiku i Rubikut*

Dedicated to my family and loved ones

ACKNOWLEDGEMENTS

I am grateful to my mentor, Assoc.Prof.Dr. Dimitrios Karras, for providing me with tremendous advice, assistance, and inspiration during this study. His skills and understanding were critical in constructing this theory.

I also want to thank the members of my advisory board for all their helpful input and ideas, which significantly increased the standard of this thesis.

I'd like to thank Epoka University for offering the tools and resources required to perform this research. Many thanks to the Department of Computer Engineering for all their administrative and technical support. My sincere gratitude goes to my coworkers and partners for their help, collaboration, and insightful conversations that enhanced this research. I am deeply thankful to my loved ones and friends for all their constant encouragement and backing. Their compassion and understanding were critical during the difficult parts of our trip.Finally, I'd want to thank everyone who participated in this study. Their contributions were critical to the conclusion of this study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

There are several methods for handling information in order to send it over the World Wide Web.

It is crucial that the mechanism used to send this data is dependable enough to avoid a data transfer loss and safe enough for preventing third parties form accessing the information. Among these techniques is cryptography, which has increased in significance in terms of offering security for many different kinds of implementations, particularly in light of the COVID-19 epidemic. According to the Central Statistics Office, "one in 10 established a web page to help with online revenues" [1] and "almost a fifth (19%) companies observed a rise in revenues via mobile apps or websites throughout the pandemic" [1].

According to the study's Internet of Things section, about 25% of these businesses would utilise these gadgets to keep their properties secure. "Compared to building companies (12%), a larger percentage of Manufacturers (26%) and Commerce (24%) companies utilise IoT gadgets for safety" [2]. The increasing use of IoT devices by various companies has increased the demand for robust cryptography methods to be implemented.

1.1 The Study

There are quite a few major issues that must be resolved when attempting to integrate a cryptography method into an Internet of Things device:

- The machine's memory and read-only memory can be extremely constrained.

  In comparison to normal PCs, there are greater restrictions on the computational capability.

- The term "real-time refers to the speed at which the gadget can react to an instruction.

IoT devices come in a vast array of varieties, but they are all rather little, making it difficult to make do with the little resources available. Certain resources, like memory, are necessary to execute a programme and hold information in memory; but, because of the insufficient processing power, processing the data may not be possible.

Given the constraints of low resources, it may not be appropriate to attempt to apply typical encryption principles to such gadgets, as quick and precise responses are required while meeting security requirements. The COVID epidemic has led to an increase in the use of IoT devices, thus it's critical that techniques for lightweight encryption be compatible with these gadgets.

One advantage of lightweight encryption is the fact that it may be applied to various kinds of devices, such personal computers, server systems, and cell phones, that lack limitations on resources that apply only to Internet of Things devices.

Even if a lot of novel methods have been developed for both conventional and Internet of Things gadgets, quantum technology may have an impact on traditional encryption. Quantum devices pose a danger to the safety of any encrypted interaction because they are capable of calculations that conventional computers are unable to. Furthermore, given quantum machines are capable of going through all potential hidden keys more quickly than every conventional computer, they're capable to crack the encryption keys that have been produced. This could make it possible for a perpetrator or eavesdropper to listen in on a conversation among two people. Quantum machines are capable of implementing Shor's and Grovers methods to crack AES and RSA. Innovative algorithms are going to be developed in response

to such danger to conventional encryption in order to keep quantum technology from jeopardising the subject's future.

1.2 Objective

Even though there have been a number of suggested encryption methods, most of these are intended for powerful computers rather than Internet of Things devices. Thus, the study's topic is: Whether an efficient algorithm be developed especially for Internet of Things devices, given the computational resources needed?

1.3 Thesis Organization

In the second part, this thesis will examine related studies. The overview of the literature will concentrate on several encryption algorithm forms, a Rubik's cube method which has some resemblance with the suggested method we will discuss, as well as safety issues connected to the Internet of Things. The AES method's encryption of information process and the operation of quantum encryption will be covered in depth within Chapter 3. The Draft Requirement of the suggested method is shown in Section 4, along with a few minor adjustments applied in the AES version to better match the Rubik's cube application. The execution of the suggested technique for decoding and encoding will be shown in Chapter 5. The stage of assessment that is presented in Chapter 6, includes a case study on the suggested method and different approaches on an internet of things device.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Historically, basic mathematical equations were used to protect messages meant for a small group of individuals, mainly for army secrecy. Although deemed safe at the time, such primitive procedures were easily analysed and broken when compared to present norms. As internet-based and digital interactions grow more prevalent, it's crucial for there to be safe procedures in place to prevent unwanted exposure of enormous amounts of information exchanged over the network. Cryptographic methods have advanced, leading to advanced study and breakdown of methods [3]. Individuals need to have faith in computer and network security in order to continue using these types of devices.

The Narrative review shall include latest research on encryption methods and security problems for Internet of Things devices. Subdivision 2.2 provides an overview of several encryption methods and their distinction. A encryption technique built on a 3x3 Rubik's cube is covered in subdivision 2.3. Given that the suggested method relies on a four-by-four Rubik's cube, it's critical to emphasise the three similarities as well as the differences. In subdivision 2.4 we are going to talk about the different types of attacks against cryptographic algorithms. The safety concerns associated with Internet of Things devices shall be covered in subdivision 2.4.

2.2. Cryptography

The Merriam-Webster dictionary defines cryptography[5] as "secret writing." Traditionally, cryptography was mostly used for secret communications. Many ancient and current encryption techniques resemble puzzles. Building and cracking the strategy depends on your ability to alter the enigma. The Substitution cypher

substitutes every character of the data to generate the encrypted text. In English, every character corresponds to a letters. One easy idea is to replace every character by a different distinctive letter, as seen from Figure 1.

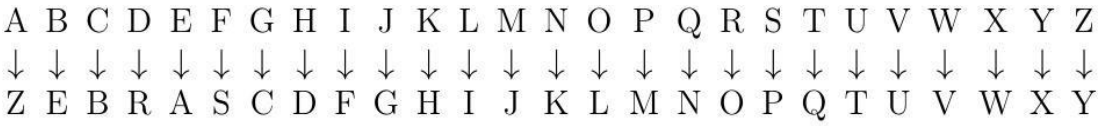


Figure 1. *Map of Characters*

2.2.1. Different Cryptographic Techniques

Encryption basic functions are categorized in 3 distinct groups: unkeyed, symmetric, and asymmetric techniques. Figure 2 summarises one of the popular encryption basic functions. Unkeyed methods employ no hidden data, making them distinct from other classifications. Symmetric techniques employ a common hidden key for encrypted processes like information encryption and decryption. Asymmetrical methods need everyone involved to have both a private key and a public one.

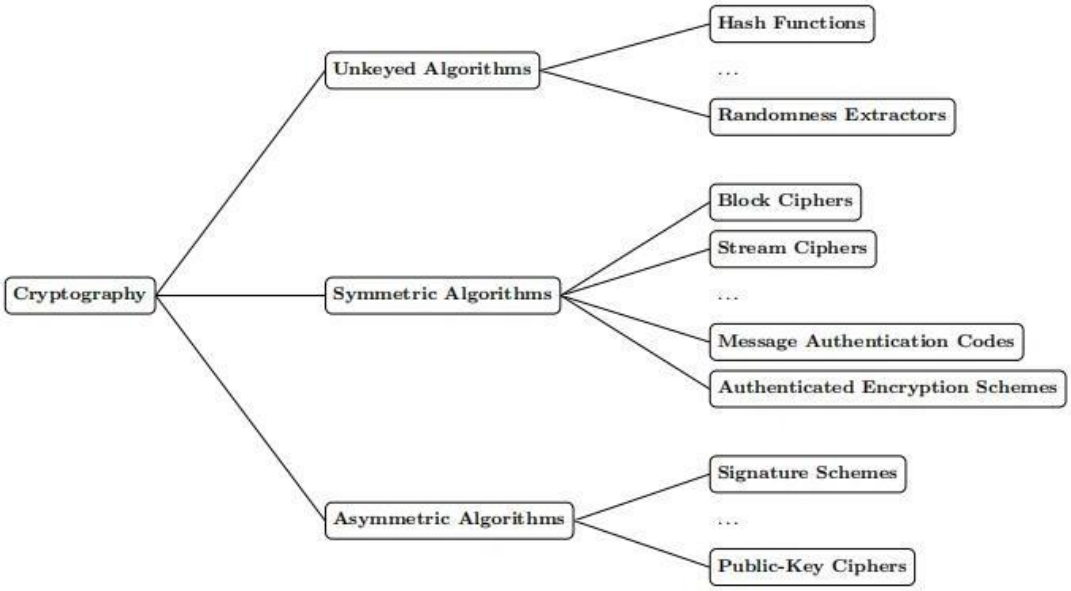


Figure 2. *Different Types of Cryptographic Algorithms*

A participant's dual keys serve distinct purposes: the visible key enables encoding and validation of digital signatures, and the secret key for deciphering as well as creating digital signatures.

Typically, protocols for cryptography are created by combining many basic functions rather than using them individually.

- **Block Ciphers**

Block cyphers play a crucial role in symmetrical encryption, ensuring information secrecy. These are widely employed to create additional cryptographic basics including stream cyphers, hash functions, and message verification codes. This section will provide an overview of block cyphers and popular building methods.

Let k, c > 1. Block cyphers are sets of tuples Π = (E, D) where the encrypting equation

$$E : F^k_2 \times F^c_2 \rightarrow F^c_2 , (K, M) \rightarrow B \qquad \text{(Equation 1)}$$

is a transformation of the collection of plain text $M \in F^c_2$ for a constant hidden key $K \in F^k_2$. The amount represented by c also additionally known as the dimensions of the block. D represents the inverse value of the encrypting equation E^{-1}, commonly known to be the decrypt equation. The formula $D_K(E_K(M)) = M$ applies to all plain text $M \in F^c_2$ and a constant hidden key $K \in F^k_2$. We identify $E_K(\cdot) = E(K, \cdot)$ as well as $D_K(\cdot) = D(K, \cdot)$, accordingly.

Conventional numbers of k tend to be 2^6, 2^7, 2^8 bits as well as 80, 96, 192 whereas b can frequently be 2^6, 2^7, 2^8 bits. Block cyphers define families of permutations.

These types of cyphers are able to encrypt one fixed-size information unit at a time. To handle data of any size, a block cypher must be employed using the correct

method of operation. NIST originally suggested and defined block cypher modes for DES in FIPS 81 as well as for AES [4]. The primary modes are: Electronic Codebook (ECB), Block Cypher Chaining (CBC), Ciphertext-Feedback (CFB), Output-Feedback (OFB), and Counter (CTR).[4]

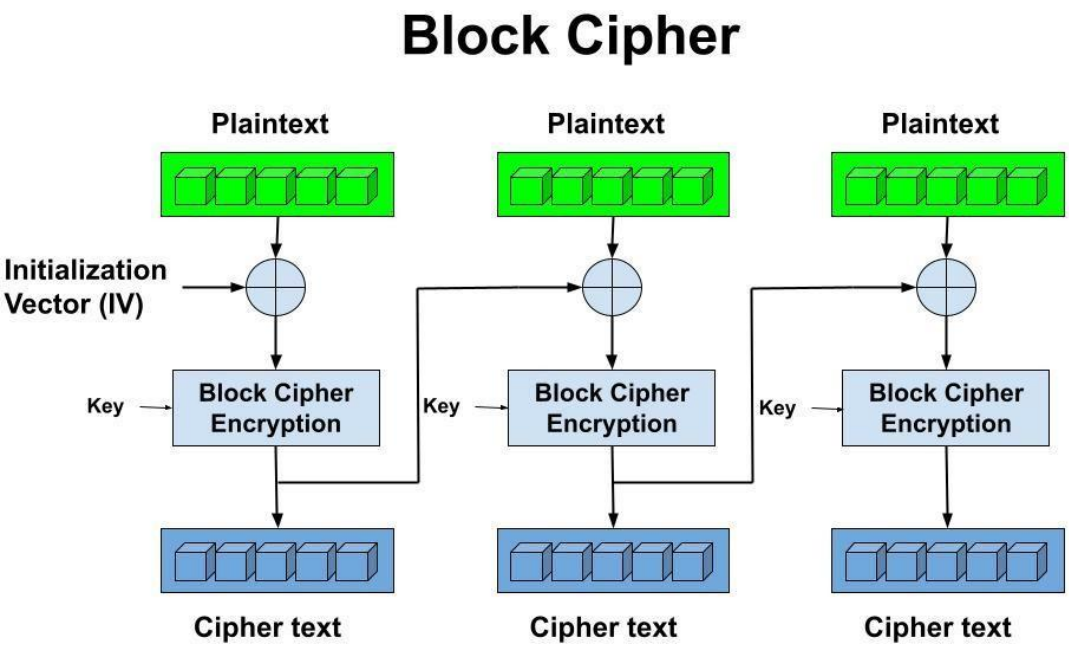


Figure 3. *Example of Block Cipher*

- **Stream Ciphers**

Stream cyphers are the second most common type of symmetric key primitives, behind block cyphers. How does it work? First to be done is the generation (almost randomly) of a flow of bits that has the exact size as the message. Secondly we use XOR on this flow of bits on the plain text to get the cypher. Stream cyphers are versatile since they do not need data padding nor specific modes of operation, allowing for immediate processing of arbitrarily length data. A block cypher may simply be converted to a stream cypher. The stream cypher S is stated as follows:

$$S: F^{k}_{2} \times F^{n}_{2} \times F^{*}_{2} \rightarrow F^{*}_{2} \rightarrow F^{*}_{2}, (K, N, M) \rightarrow S \oplus M \qquad \text{(Equation 2)}$$

in which K represents a hidden key, N is a vector of initialization, M is the data, and S is a constructed key flow of size |M|. The encrypted text C represents the result of S $\oplus$ M of S. XOR, as an involution, may also be utilised during decryption by swapping C and M positions. To obtain the original text, just compute S(K, N, C) = S $\oplus$ C = M. RC4, created by Rivest in 1987, remains a renowned stream cypher that continues to be widely utilised despite vulnerabilities and may be exploited for realistic assaults.

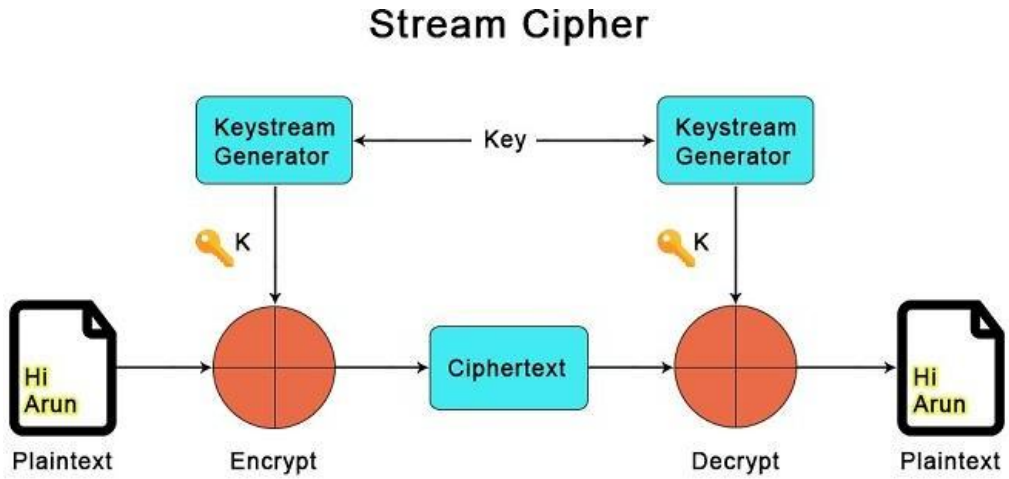


Figure 4. *Example of a Stream Cipher*

- **Hash Functions**

Cryptographic hashing algorithms maintain the confidentiality of the information being processed. These serve as building blocks for stream cyphers, data authentication codes, and encrypted authentication systems.

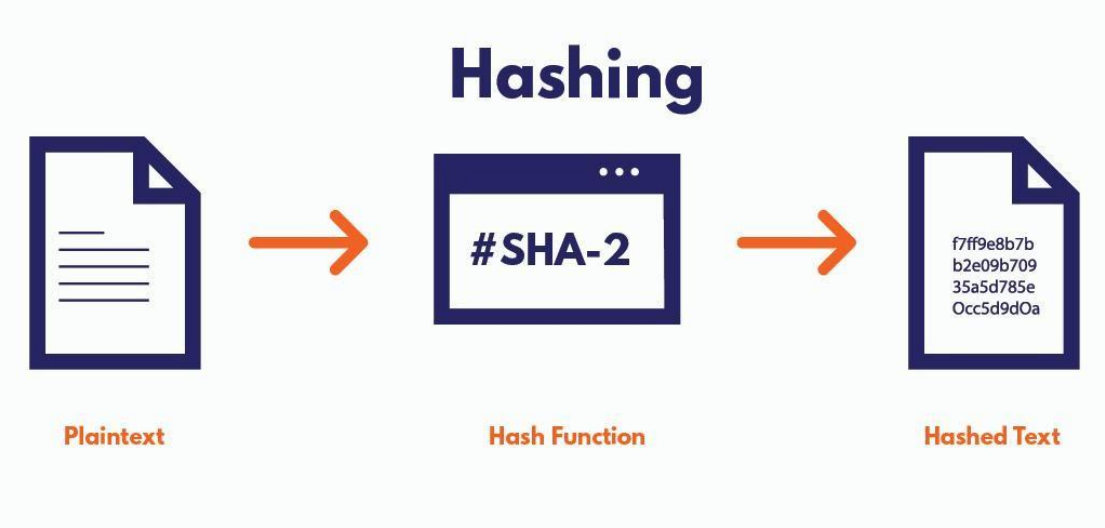


Figure 5. *Hash Functions*

Despite not addressed in full throughout the thesis, we lightly explore them here to ensure clarity. In contrast to previous symmetric primitives, hash algorithms condense an input with a length that is fixed to an output without using a secret key. The above mentioned functions are classified as one-way operations, making inversion nearly difficult. Hash functions are known for their variety and utility. Their range of applications include checks of integrity for the information, password authentication, random number creation, and message authentication. A function that generates a hash is defined in the following way:

We take n ≥ 1. A function that generates a hash compresses information (M) of indeterminate length into a digest (H) of fixed length n.

Cryptographic hash algorithms ought to resemble randomised functions.

$$H : \mathrm{F}^{*}{}_{2} \rightarrow \mathrm{F}^{n}{}_{2},\ M \rightarrow H \qquad \text{(Equation 3)}$$

2.2.2. Comparison of different Cryptographic Algorithms

We compared the safety of regular and lightweight techniques. Performance is a crucial aspect of cryptography algorithms. The technique known as AES outperformed DES and RSA in terms protection due to its ability to employ different key sizes. Several attempts had been proposed in order to break the AES, including square, key, and differential attacks, however nothing was successful [6]. DES lacked sufficient security due to its keys with size of 56 bit. "Brute force attack becomes possible with a massively parallel machine of more than 2000 nodes with each node, capable of a key search rate of 50 million keys/sec" [6]. DES's small lengths of keys as well as weak S-boxes make it vulnerable to attacks. The RSA technique's integrity depends upon the difficulty of factoring huge integers. The public key comprises of two integers, one of which is the product of a pair of prime numbers. If such number gets factored, the secret key will be revealed. Although tedious, RSA has shown to be a robust technique. Rating each algorithm by safety, AES is the safest and DES the least secure.

Lightweight cryptographic algorithms like PRESENT and SIMON are well-suited for internet of things devices, providing both decryption and encryption capabilities.

PRESENT may work using a 64-bit block, while the key length may vary between 80 to 128. Research has proved that using biclique cryptography on the version with 80 "uses 527 Sboxes in a full-round encoding, thus the computational cost corresponds to the following equation:" [7]

$$2^{72}\left(\frac{84504 + 633 + 137}{527} + 0.0625\right) = 2^{79.34} \qquad \text{(Equation 4)}$$

SIMON came out in June 2013 and has been adjusted for hardware efficiency. The Simon block cypher uses a block with size 2^n-bit as well as a mn-bit key, in which n is 16, 24, 32, 48, or 64 bit and m is 2, 3, or 4 bit. Simon32/64 is a variant of Simon that uses a 64-bit key and works with 32-bit text chunks [6]. Comparing such algorithms, it was found that they performed well in regard to technology, power, as well as space utilisation. Although these methods are appropriate for the Internet of Things data encryption and decryption, threats remain conceivable, as seen in Figure

5. While AES is an extremely safe and efficient technique, PRESENT and SIMON additionally offer sufficient safety for the Internet of Things due to their minimal computing resource requirements.

Table 1. *Algorithm Comparison Table*

Ref.	Algorithm	Key Size (Bits) Block Size (Bits) Rounds			Structure	Performance				Merits	Attacks/ Analysis
						Tech. (µM)	Power (µW)	Area (GE)	Throughput At 100Khz (Kbps)		
[17]	AES	128	128	10	SPN	0.13	2.48	2400	56.64	Supports larger key sizes, faster in both hardware and software	Related key attack, Boomeran, Biclique cryptanalysis
[19]	PRESENT	80	64	32	SPN	0.18	1.54	1030	12.4	Ultra Lightweight cipher, Energy efficient.	Integral, Bottleneck attacks, truncated differential cryptanalysis, Side-channel attacks
		128				0.18	2.00	1339	12.12		
[20]	RECTANGLE	128	64	26	SPN	0.13	1.78	1787	246	Fast implementations using bit slice techniques	Slide attack, related-key cryptanalysis, statistical Saturation Attack
[22]	HIGHT	128	64	32	FN	0.25	5.48	3048	188.20	Ultra-lightweight, provides high security,	Impossible differential attack on 26th round,

										good for RFID tagging	Biclique cryptanalysis
[23]	CLEFIA	128	128	18	FN	0.13	2.48	2488	39	Has fast encryption and decryption, lesser rounds, efficient energy	Key Recovery Attack on 10th round, Saturation Cryptanalysis
[26]	CAMELLIA	128	128	18.24	SPN	-	1.54	6511	290.1	Resistance to brute force attack on keys, security levels comparable to AES	Cache timing attacks, Impossible differential attack
[28]	TWINE	80.128	64	36	FN	0.09	1.30	1866	178	Good for small hardware, efficient software performance	Meet-in-the-middle attacks, Saturation Attack
[29]	SIMON	128	128	64	SPN	0.13	1.32	1317	22.9	Supports several key sizes, performs well in Hardware	Differential fault attacks, Attacks on reduced versions
[29]	SPECK	128	128	32	SPN	0.13	1.40	1396	12.1	Performs better in software	Key Recovery, Boomerang attack

2.3. Rubik's Cube Cipher

On the ninth of May 2010, a study proposed an encryption method using a three-by-three Rubik's cube. The novel approach addresses flaws within the Vigenère & Mitchell methods, preventing assaults via anagramming as well as analysis of frequencies.

Vigenère Cypher operates as a polyalphabetic shifting cypher, meaning that each text character may redirect to various encrypted text characters determined by a keyword or shifting. Assuming the text in question is MASTERTHESIS and the key will be STUDY, it will expand to match the text's size. According to Table 2, our text should look as follows.

Table 2. *Demonstration Of Viangère*

M	A	S	T	E	R	T	H	E	S	I	S
S	T	U	D	Y	S	T	U	D	Y	S	T
E	T	M	W	C	J	L	B	G	Q	A	L

To generate a cypher turn the text as well as keywords onto matrices with integers assigned to the letters of the alphabet. Consider the term STUDY as an array of values 0-25 translated into the alphabetical order in an identical sequence (A = 0, B = 1, etc.). The keyword will be translated into the subsequent array (18, 19, 20, 3, 24). Applying the identical procedure to the primary five characters within the unencrypted text results in the array (12, 0, 18, 19, 4). Table 2 shows that employing unencrypted text M as well as key S results in encrypted text E, whereas utilising transformed matrix variables 18 and 12 yields ciphertext vector value 5.

A regular 3x3 Rubik's cube has 54 fragments, all of which have six sides plus 9 cubies. Mitchell's approach includes modifying the unencrypted text prior to rearranging it. The initial procedure involves writing "1" onto the uppermost left quadrant of a cube side. The value "2" may be placed in an arbitrary cube on any other side, and this continues till every one of the faces possess a unique identification [8]. To completely fill the cube's surface, begin with the side containing the value 1 and populate all blank cubies with unencrypted text letter. Repeat for the side containing the value 2 till every cubie is filled. Figure 6 illustrates how it is done using the following unencrypted text value:

LEARNING CRYPTOGRAPHY INNCI HAS BEEN A GREAT EXPERIENCE

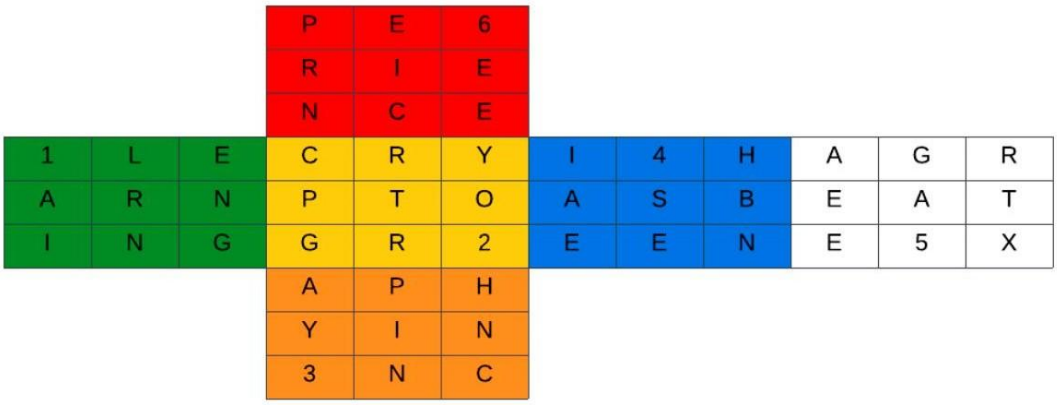


Figure 6. *The First Step Of The Algorithm*

The next step in the procedure was to produce a rotating key. Such key allows the cube to return to its original configuration from the initial stage. The upper leftmost corner of any selected side has to start with a value of 1. The rotating key allows for about 7.25 billion encryptions based on how the unencrypted text has been initialised. Insufficient unencrypted text repetition prevents attacks from succeeding. Numerous anagramming attacks are problematic as they rely on using the identical inversion ordering again [8].

Mitchell proposed a scheme for allocating cube sides using a six-letter addition, such as "ACEDBF". This particular rule instructs the party that received it on how to allocate the encrypted text to the cube. To shuffle the cube, use R = Row, C = Column, and L = Level. The digits 1, 2, and 3 denote the clockwise movement in multiples of angles of ninety degrees, whereas 4, 5, and 6 denote spin of the cube's outer two layers in multiples of angles of ninety degrees. Figure 7 illustrates a scenario of a rotating key based on the above specifications.

AECBFDX-L4-C6-R1-C5-L3-R2

Figure 7. *Example For The Key Used In Rotation*

Using a Rubik's cube for encryption might lead to repetition of letters throughout the cypher text, reducing the key value. "Research has demonstrated that God's Number is number 20" [9]. God's quantity to solve a three by three Rubik's Cube refers to the greatest quantity of movements required for solving any variation of the "43 quintillions" combinations of the cube [9].

2.4. Cryptographic Attacks

In order to check the safety of a cryptographic algorithm the need to use various attacks arose. In this part, we will discuss some of the different types of them.

2.4.1. Categories of Attacks

To determine how efficient a attack is we use three parameters: how time consuming the attack is, how much memory it uses and the data. An attack's success is often judged by its time, memory, and data requirements. The result produced by an adversary's assault is determined by two elements: the objectives that they aim to attain as well as the adversarial model that limits their actions. A generic approach on encryption structures is one which operates not relying on specific data about the class members. A generic assault involves extensively examining all key possibilities for a symmetric-key primitive. Non-generic attacks are those that need certain encryption building elements. In the analysis of the attacks, usually we think that the attacker has complete knowledge about the targeted encryption primitive,

excluding the user-supplied hidden key. Kerckhoffs' Principle, established by Auguste Kerckhoffs in 1883 [10], defines the prerequisites to establish a viable field cypher.

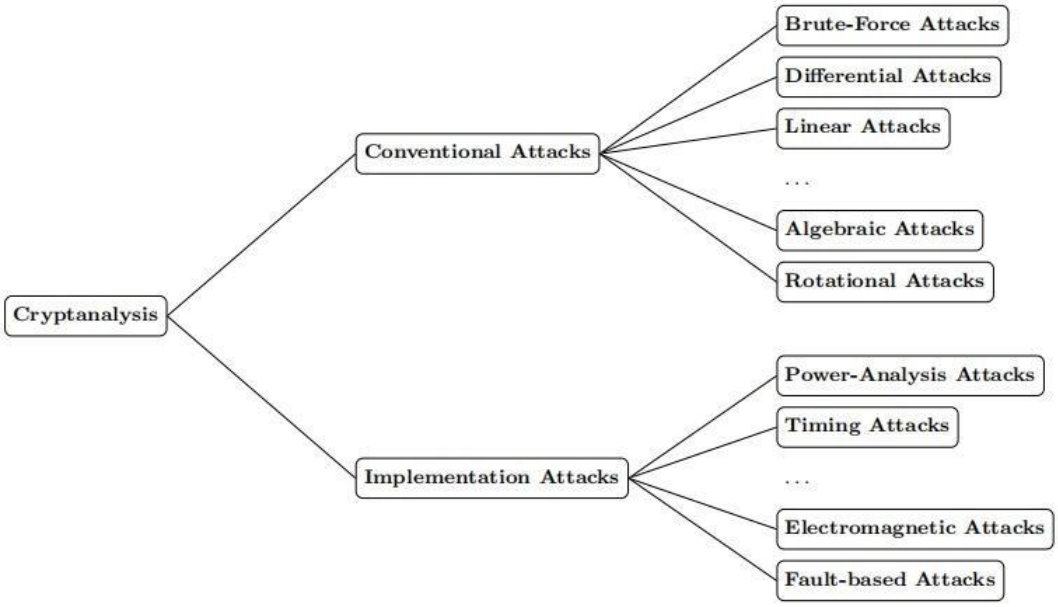


Figure 8. *Categories of Attacks*

2.4.2. Objectives of an Attack

In this section we are going to give a simplified categorisation of the objectives of the attacks.

1. Key Recovery. The intruder may retrieve the hidden key K. This can be probably the most effective outcome of an assault.
2. Global deduction. An intruder can determine encrypting (EK) or decrypt (DK) while not obtaining the hidden key (K).
3. Local deduction. An intruder may carry out encrypting (EK(M)) or decrypting (DK(C)) despite obtaining the key (K) for particular transmissions or

encrypted texts.

4. Differentiating. An intruder may differentiate between EK(·) and randomly generated permutations. The simplest attack against an encryption primitive is attempting to identify encrypted information from data that is random.

It is to be noted that the ranking described above is specific for block cypher primitives.

2.4.3. Brute-Force Attacks

A basic approach to most symmetrical cryptographic primitive involves searching exhaustively to locate the shared hidden K. This technique has no connection to of cypher architecture. In block cyphers, the attacker evaluates every possible key options versus an identified message-ciphertext combination to find the proper key. This type of cryptanalytic approach is additionally referred to as brute-force assaults. Encryption primitive architects want to make brute-force the most effective attack accessible to adversaries, knowing that there is no mechanism to avoid such extensive searches. Sophisticated cryptanalytic assaults frequently include extensive methods of search.

An intruder having knowledge of a ciphertext combination with the message (M, EK(M)) as well as the encrypting method, may discover the hidden key K having a chance by trying 2^k keys, where the key equals |K|. Overall, if $n \leq 2^k$ tests of keys are made, he is successful having a chance of $n/2^k$. Assuming he checks $m < n/2^k$ simultaneously, the likelihood is $mn/2^k$.

2.4.4. Differential Attacks

Biham and Shamir established differential cryptanalysis during the beginning of the 1990s [11] by investigating threats on block cyphers as well as hash algorithms.

Researchers found that DES is highly resistant to many assaults, but with simple adjustments, it may become less secure.

Coppersmith, one of the initial DES creators, presented a study in 1994 which revealed the IBM design group was conscious of differential cryptanalysis as far back as 1974 [12]. Differential cryptanalysis is a helpful instrument for cryptanalysts. Originally developed for block encryption [14], it has now been applied to various symmetric primitives [13].

Differential attacks employ connections among both input and output variations within an encryption primitive. They use non-ideal transmission of variations between unencrypted and encrypted text pairings. Variations are often calculated using bit-wise XOR, however they are also an option in other circumstances, such as modular numeric addition. Differential cryptanalysis falls within the domain of unencrypted and encrypted text assaults, as previously mentioned.

A basic encryption system, comparable to an only once pad, encodes a text M using a key K and generates encrypted text C via calculating $C = M \oplus K$. If K is utilised again for encryption of a different message M0 ($C_0 = M_0 \oplus K$), an intruder that captures simultaneously C along with C_0 may easily determine the plain texts via calculating its XOR-difference between the unencrypted texts.

$$C \oplus C' = (M \oplus K) \oplus (M' \oplus K) = M \oplus M' \qquad \text{(Equation 5)}$$

This short sample effectively demonstrates the fundamentals of differential cryptography analysis. To analyse actual cyphers, a broader method to differential cryptography is necessary due to their complexity.

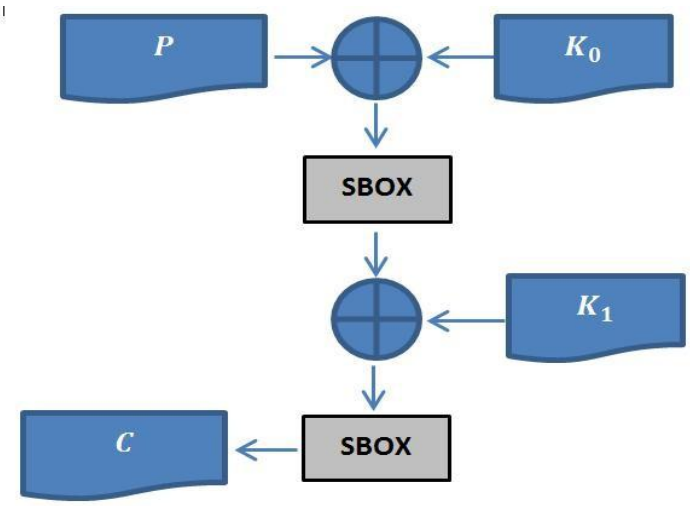


Figure 9. *Differential Cryptanalysis*

2.4.4. Linear Attacks

Matsui proposed linear cryptanalysis around 1992 for targeting a block cypher (FEAL) [18] and then expanded it to DES [18]. The latter is one of the earliest widely publicised cryptanalysis findings for DES. Following differential cryptanalysis, linear cryptanalysis is the next significant approach for analysing encryption primitives. As previously stated, it comes with the benefit of just needing known unencrypted text rather than selected plaintexts compared to differential attacks.

Linear cryptography uses formulas to describe key bits such as text and encrypted text bits, with not linear cypher elements represented by linear estimates. The intruder aims to create a linear approximate value over the very first r − 1 phases of the block cypher ($f_{r-2} \circ \cdots \circ f_0$) that ensures $b_0 \cdot y_0 \oplus b_{r-1} \cdot y_{r-1} = 0$, in which b_0 as well as b_{r-1} tend to be linear masks regarding the inputs and results after r − 1 rounds. The dot product of F_2 will be indicated by ·. For two components, $y = (y_0,..., y_{n-1})$ with $x = (x_0,..., x_{n-1}) \in F^n_2$, it can be expressed with $x \cdot y = \bigoplus^{n-1}_{i=0} y_i x_i$.

The chance of an approximate linearity is described with $p = 1/2 + \varepsilon$, in which ε indicates bias. Assuming $f_{r-2} \circ \cdots \circ f_0$ represents a perfect block cypher, therefore $\varepsilon = 0$, implying that the formula before applies with likelihood 1/2. The linear approximate can differentiate $f_{r-2} \circ \cdots \circ f_0$ compared to a perfect cypher when its bias ε varies substantially compared to 0 as well as p diverges substantially from 1/2. This involves encryption of plaintexts x_0 at arbitrarily to results x_{r-1} and analysing the frequency that represents the linear connection among them. The intruder benefits from a bigger $|\varepsilon|$ value, which may be either positive or negative (often $0 < |\varepsilon| \leq 1$).

When a distinguisher is discovered, it may be used to perform a key recuperation assault upon the full block cypher $f_{r-1} \circ \cdots \circ f_0$. This involves assuming the bits x_{r-1} in the encrypted text x_r using the result mask α_{r-1}. With sufficient unencrypted and encrypted text combinations, the key option with the largest biases is probable to be the right one.

Linear likelihood, features, and hulls are defined similarly to differential chances, characteristics, as well as differentials [12]. The piling-up lemma [12] is a useful tool for calculating the linear likelihood of a linear hull based upon its linear traits. It suggests the likelihood p of the total of m (unrelated) Boolean equations with likelihoods pi for $i \in \{0,..., m - 1\}$ could be calculated using the equation below.

$$p = \frac{1}{2} + 2^{m-1} \prod_{i=0}^{m-1} \left(p_i - \frac{1}{2} \right) \qquad \text{(Equation 6)}$$

In linear cryptography, matrices of correlation are commonly employed to express Boolean equations [15].

In [16], the achievement likelihood of differential attacks against a thorough examination is analysed, as well as that of linear assaults. Assume p_s is the rate of successfully attacking, the number of accessible unencrypted and encrypted text combinations is denoted with N, while the outsider seeks an edge worth m bits above exhaustive searching. Then we will get the following:

$$p_s = \Phi \left(2\varepsilon\sqrt{N} - \Phi^{-1}(1 - 2^{-m-1}) \right) \qquad \text{(Equation 7)}$$

Assuming the likelihood of an approximate linear model is distinct for every key option and equivalent to 1/2 for incorrect predictions, as well as m and N are suitably big. This technique may be modified to estimate the data demand N based on a particular successful probabilities (ps).

$$N = \left(\frac{\Phi^{-1}(p_s) + \Phi^{-1}(1 - 2^{-m-1})}{2} \right)^2 \varepsilon^{-2} \, . \qquad \text{(Equation 8)}$$

In simple terms, complexity of information correlates to 1/ε2. Experiments show that linear attacks calculations tend to be more exact than differential variants, indicating reasonable hypotheses. Bogdanov and Tischhauser improved their successful probability calculation, reducing the complexity of data for linear assaults [17].

While linear attack lacks the versatility of differential attack, it may be extended in multiple directions. Using many linear estimates simultaneously can

minimise data requirements [18]. Knudsen demonstrated that using chosen unencrypted text can lessen the difficulty of linear attacks against DES [19]. Knudsen's non-linear estimate approach to linear attacks can yield further details [19]. Bogdanov and Rijmen introduced zero-correlation methods in 2011 as an alternative to inconceivable differentials in linear cryptanalysis. These attacks use linear estimates with probability of precisely 1/2.

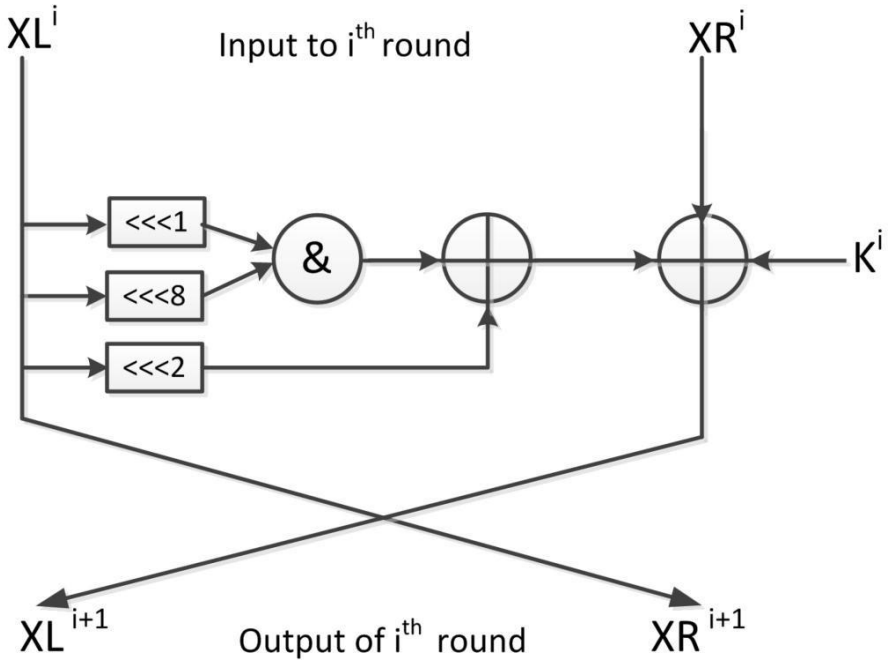


Figure 10. *Linear Attacks*

2.4.5. Algebraic Attacks

Breaking a cypher employing this type of attack often involves two phases: To simulate the cypher procedures algebraically, use a set of non-linear multidimensional polynomial $f_1,..., f_s$ P. In this instance, $P = K[x_1,..., x_n]$ represents a polynomial circle in which n determines $x_1,..., x_n$ on a domain K. In symmetric encryption, K is often the limited F^2_r in feature 2 (r 1). Calculate this set of function S with

$$\begin{cases} f_1(x_1,\ldots, x_n) = 0 \\ \quad . \\ \quad . \\ f_s(x_1,\ldots, x_n) = 0. \end{cases} \qquad \text{(Equation 9)}$$

the essential variables, which breaks the cypher. Computing multi-modal equations that are not linear on limited fields is NP-hard, despite its seemingly straightforward idea. Nevertheless, there are still occasions when answers may be quickly retrieved, despite the overall complexities. For instance, algebraic attacks were used to effectively crack the cypher Crypto-1 [20]. Algebraic approaches are very effective when combined with side-channel assaults, and are commonly utilised in this scenario.

Constructing a cypher using polynomials f1,..., fs may be performed in several ways, with differing representation potentially affecting the solution time of the system's. Polynomial systems may often be stated solely in terms of key indeterminate elements. Yet, the resultant polynomials frequently include a high degree and many monomials, making them unsuitable for the system solution. Often, more indeterminates as well as formulas are thrown in. This ends up with an over defined structure, with more formulas than parameters.

To ensure whether all answers obtained using the algorithm are within the mathematical closure F2 r, the domain formulas x 2 i r - xi = 0 for i 1,..., n are often added to S. There are two techniques for analysing a system of formulas:

- **Gröbner Bases**
- **SAT-Solvers**

We are not going to cover these solvers on our thesis.

2.4.6. Rotational Attacks

Khovratovich and Nikolić [21] presented rotational attacks to analyse ARX-based encryption primitives. ARX, which stands for modular number addition, bit-wise rotation, as well as XOR, is a typical technique for designing algorithms for encryption. Instances of these primitives are BLAKE(2), ChaCha, Salsa20, Skein, or Speck. Rotational encryption involves tracking the spread of rotational connections via cryptographic transformations. Detecting rotation-invariant conduct allows for the creation of distinguishers and key retrieval assaults, comparable to differential tactics. Rotational attack has been effectively used to lower variants of cryptographic fundamentals, such as Skein along with Keccak.

2.5 Internet Of Things

In this section of the thesis we are going to talk about Internet of Things (IOT). We are going to discuss what IOT is, what type of devices are included in IOT, a simple design of a IOT as well as some protocols.

2.5.1 What is IOT?

Procter & Gamble's Kevin Ashton coined the expression "Internet of Things" in the year 1999. The IoT refers to a collection of networked contents that use software, electronics, and various types of technology to exchange information among other gadgets and systems via the worldwide web. Such items might range from modest domestic gadgets via sophisticated commercial and industrial apparatus. Figure 12 depicts the interaction between humans, objects/devices, and the internet that creates the IoT ecosystem.

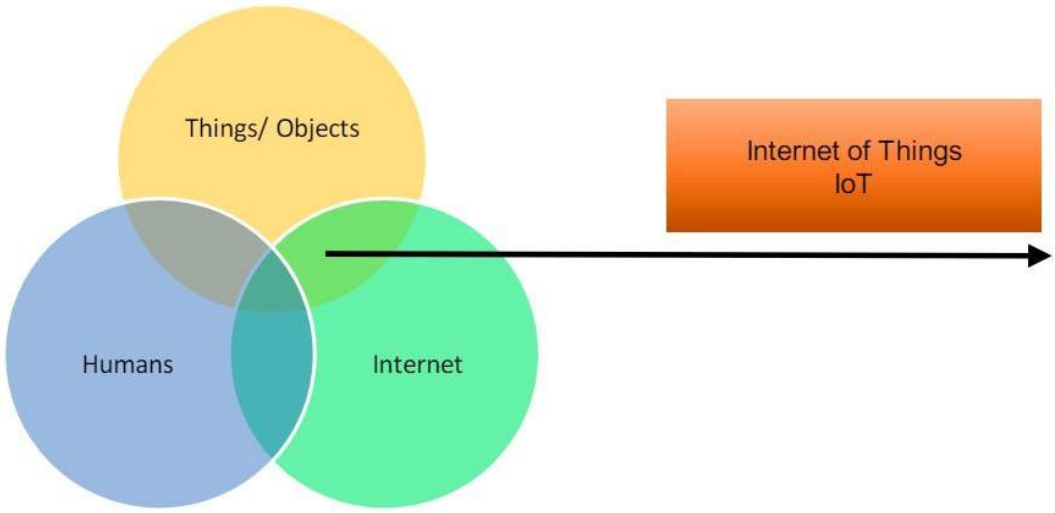


Figure 11. *IOT "ecosystem"*

Stojkoska and Trivodaliev (2017) state that smart objects capable of communication and computation surround us, including simple sensors, home appliances, advanced smartphones, and industrial devices. The notion of IoT encompasses these different networks of items and gadgets. According to Vermesan/Friess (2013), the Internet of Things is an infrastructure that allows diverse items to join at any time and from any location over the World Wide Web.

2.5.2. IOT Devices

According to Radoglou Grammatikis, Sarigiannidis, and Moscholios (2019), the IOT is made up of multiple systems that allow objects to connect with one another over the World Wide Web. Figure 12 illustrates these gadgets, sometimes known as 'things'. Each 'object' has a completely distinctive collection of characteristics.

1. **Identification**: The fundamental need for internet of things gadgets is that they must be individually identified within the internet of things. Network entities are assigned distinctive addresses using one of two techniques: Internet Protocol version 4 or Internet Protocol version 6.
2. **Sensing:** It entails gathering data gathered from the world around us. Different sensing devices, including smart sensors.

3. **Communication**: This process entails the transmission and reception of data, messages, files, and other types of information through connected devices. Bluetooth connectivity, internet connections, Radio Frequency Identification, among other technologies help items communicate with one another.
4. **Computation**: Computation serves to handle the data acquired from IoT devices. This procedure also serves to remove redundant or surplus data.
5. **Services**: Services denote the functionalities of devices provided for customers according to the data they get.
6. **Semantics**: This is the ultimate feature of internet of things gadgets. It refers to their capacity to collect precise data in the real world and offer the information as an asset at the right moment or when needed.

Figure 12. *IOT device atributes*

2.5.3. How does it work?

The internet has revolutionized the world, transforming how we work and communicate with each other. This evolution will persist after the emergence of new innovations like fifth-generation novel protocols for the internet like LiFi. IoT changed communication by allowing several objects to connect to the internet at the same time. This development not just improves relationships between individuals and machinery, it additionally also promotes communication among machines themselves (Stojkoska and Trivodaliev, 2017). Such capability has unlocked boundless opportunities for exploration and utilization, both on personal and business fronts.

Devices share data through an IoT gateway, functioning as a central hub for all connected technology. Additionally, an edge device can be integrated into the system to analyze data locally, reducing bandwidth by transmitting only the filtered information. However, the relationship isn't always strictly device-to-gateway. Smart gadgets on the same network (Wi-Fi, Bluetooth) can interact and act on the data independently. Some even leverage AI and machine learning for optimal efficiency.

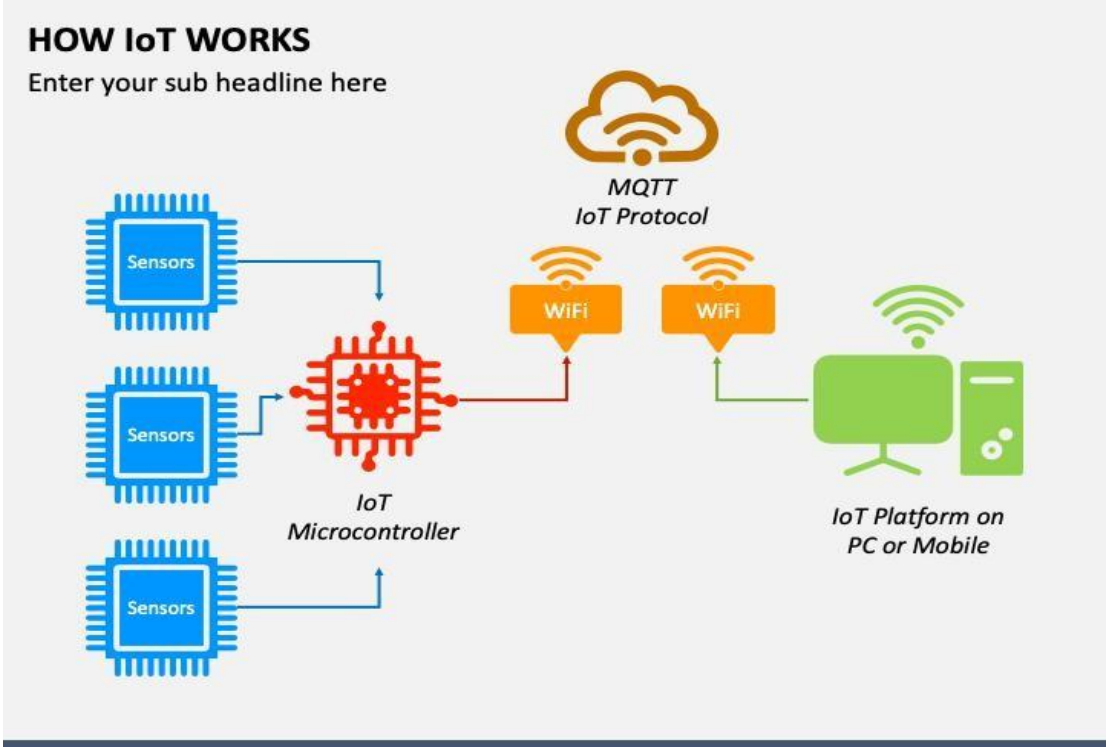


Figure 13. *Example of how IOT works*

2.5.4. Security in IOT Systems

Security is paramount for establishing Confidential and secure interaction among persons, applications, and various other things. This involves a triad of processes:

1) **Availability:** To facilitate data exchange between two different individuals or parties, it is essential to identify and authenticate both sides. The challenge in the

IoT realm is that, besides its users, other services might interact with the device in use. To ensure secure authentication between different devices or parties, a pre-defined procedure must be in place before any interaction occurs. These systems can be configured to activate when the primary system is disrupted or fails.[22]

2) Integrity: Ensure the information transferred among persons or organisations remains untampered is crucial. The accuracy of the transmitted data must be perfect; any deviation compromises its integrity. The accuracy of data exchanged between IoT devices can be impacted due to their low computing power. Although IoT must adhere to conventional safety techniques, these are often not followed because of limited resources.

3) **Confidentiality:** Even if data integrity is maintained and there are standard procedures between parties, the data is considered public rather than private if it can be accessed by an unauthorized user. To ensure private information remains secure, an authorization system must be in place to control who has access to necessary privileges. Since IoT users include both humans and various services, it is crucial to protect data throughout its life cycle to maintain confidentiality.

While the CIA triad is crucial for security, the IoT requires two additional processes: Efficiency in energy use and variability. Because every IOT gadget runs on power from the battery as well as self-harvested power, there is a need to minimise the use of energy so that every gadget operates efficiently.[23] Failure to reduce energy usage can shorten a device's lifecycle, impacting the IoT network. Given that IoT devices can integrate heterogeneous devices, a diverse array of devices can be connected to a connection. Measures regarding the safety must be used to manage this variety of heterogeneous gadgets effectively.

In the area of encryption, it is essential to have algorithms that use energy efficiently. In 2013, the NSA received numerous proposals for compact block cyphers. The pair of primary algorithms proposed were SIMON and SPECK, however both were criticised by numerous security specialists and declined by International Organization for Standardization in 2015.

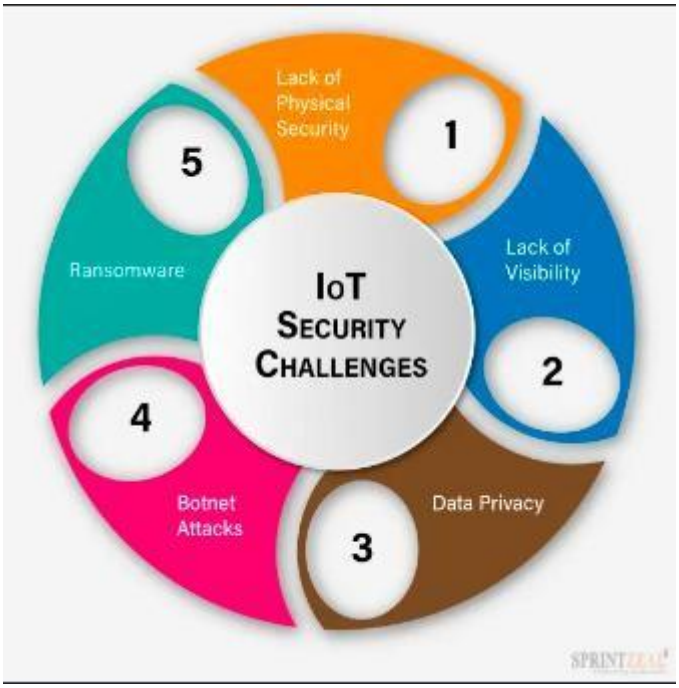


Figure 14. *IOT Security Challenges*

2.5.5. IOT Architecture Layers

Figure 15 depicts three separate levels of the Internet of Things.Unfortunately, every one of these levels includes weaknesses in security that might compromise encrypted data.

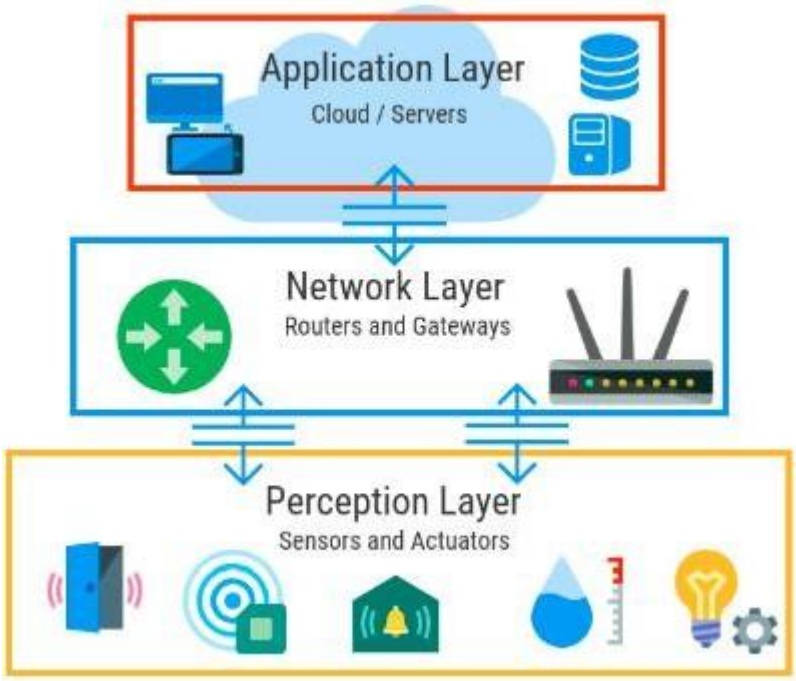


Figure 15. *IOT Architecture Layers*

- **Perception Layer**

This level may include intelligent devices such as monitors or microcontrollers. These gadgets' strong computational capacity allows them to be linked to the the internet, and this in turn connects to the programme operating. Because this level collects information, its essential to ensure the information is authenticated by the gadget prior to being delivered over the network level. This type of layer has a couple safety problems: node manipulation and harmful code insertion. Interfering with the network nodes may give a hacker access to sensitive information, including the encryption keys for an encryption method. If equipment remains outdated, an intruder may be able to insert code and obtain entry to the connection.

- **Network Layer**

Data within the perception level is transported throughout the internet, which includes the layer of network. This section appears more vulnerable than the layer of perception since data may be transferred from a wide range of gadgets. This level is vulnerable to a variety of assaults, such as routing, distributed denial of service and man-in-the-middle. Attacks on routing change the course of information transfer, possibly jeopardising its confidentiality. Considering the heterogeneity of today's technologies, each rogue machine may launch a DDoS assault and damage the connection. Man-in-the-middle attacks are possible if gadgets that transmit and receive information aren't properly protected.

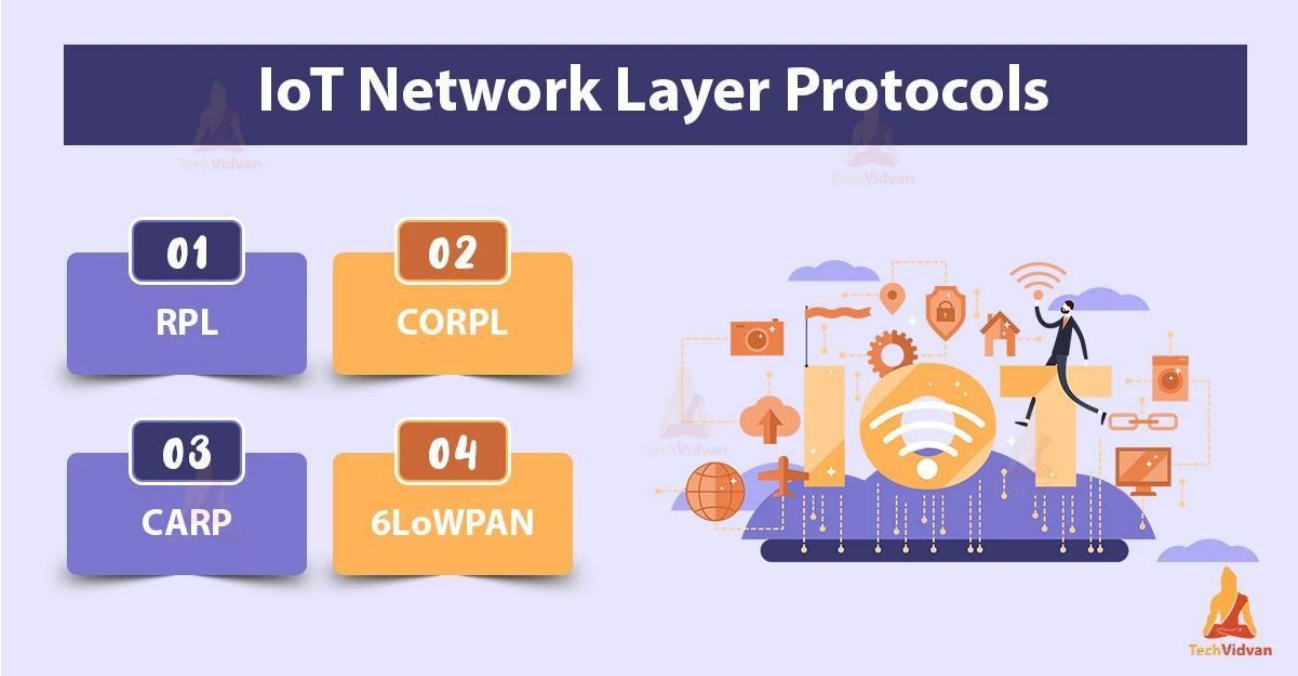


Figure 16. *Network Layer Protocols*

- **Application Layer**

It gets information via the network side and utilises it to carry out the operations for the Internet of Things system. This layer constraints involve handling access to data rights and identification verification., particularly given the diversity

of applications and users involved. Threats at this level include data leakage and device misconfiguration. Data leakage poses a security risk if attackers exploit application vulnerabilities, potentially leading to data manipulation stored on the cloud. Moreover, incorrect security configuration of IoT devices, such as operating systems or databases, can render the device vulnerable to attacks, potentially compromising the entire IoT network.

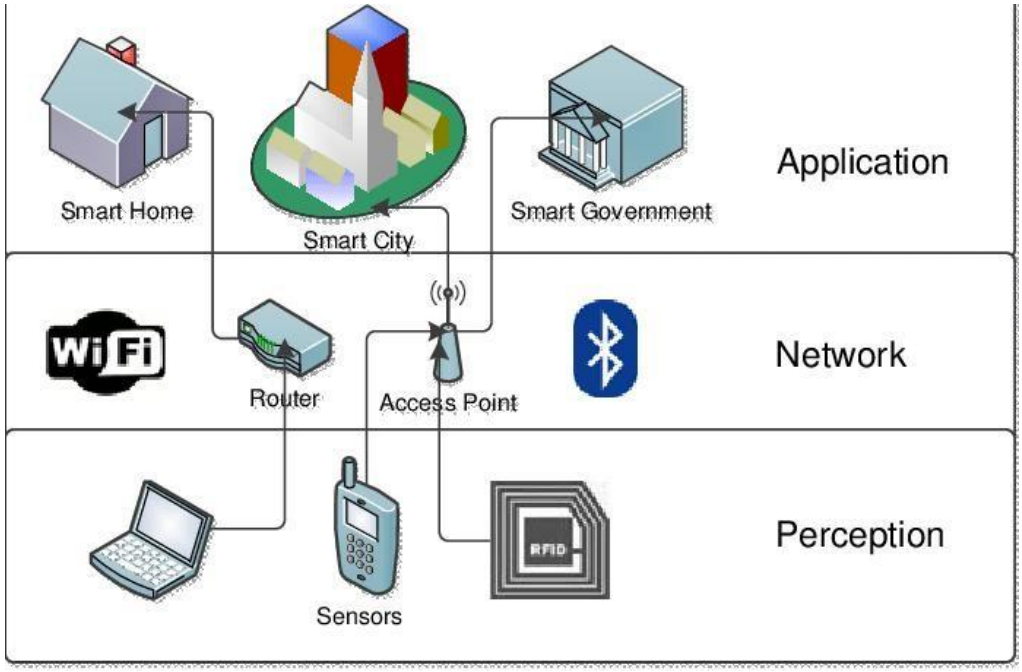


Figure 17. *Another IOT Architecture Example*

CHAPTER 3

METHODOLOGY

3.1. Introduction

According to a recent study, which identified weaknesses in security throughout all levels of IoT, novel encryption techniques are required to resist the danger of quantum technology. It is difficult to provide efficient safety on IoT gadgets due to their minimal computational capacity. The suggested approach intends to improve Rubik's Cube rotating generated keys by using the algorithm developed by AES to build a strong cypher. The following paragraphs will cover the algorithm known as AES as well as offer a summary of Quantum Cryptography, highlighting its effectiveness and rapidity in comparison to conventional encryption techniques.

3.2. Advanced Encryption Standard Algorithm

As discussed previously the AES algorithm, or the Advanced Cryptography Standard is part of a block cypher encryption submitted to the NIST during the 2000. This had been developed to substitute the algorithm known as DES due to discovered safety problems. AES is currently regarded as the most secure algorithm available. Secure and interpret data after the encryption, AES divides the information into 16 bytes as well as operates using on a matrix. AES supports sizes of keys of 128, 192, or 256 bits, with the algorithm performing actions in given the key measurements, the rounds might be 10, 12, or 14. Figure 19 illustrates the steps taken in each round.

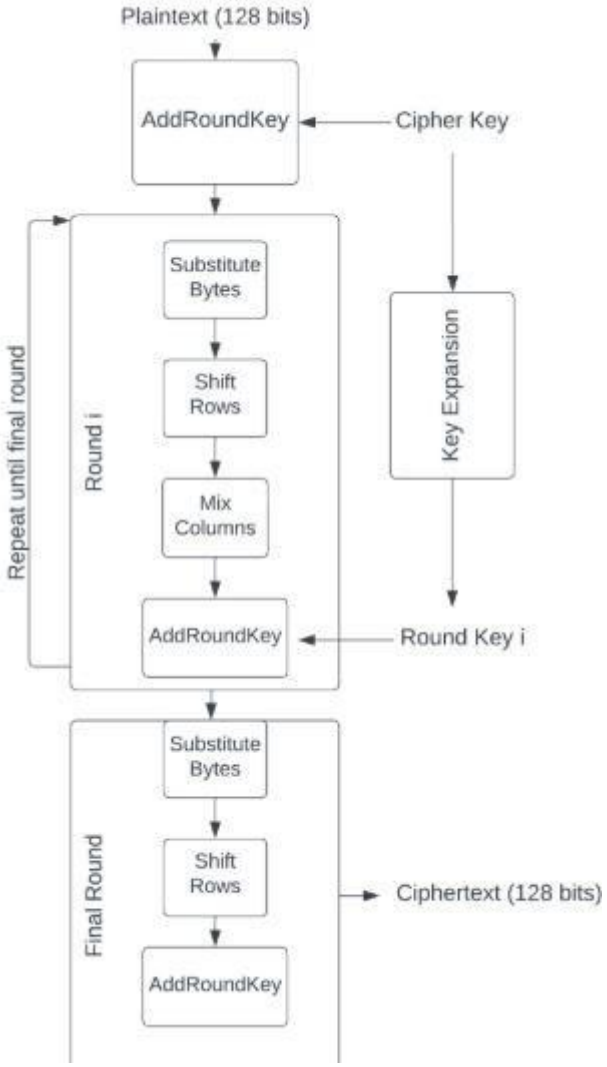


Figure 18. *Advanced Encryption Standard Architecture*

- **ShiftRows**:As the name suggests, this step involves shifting each row a certain number of times:
    - The initial line remains unchanged.
    - The next line shifts left one time.
    - The next line shifts left two times.
    - The last line shifts left thrice.

$$\begin{matrix} a0 & a1 & a2 & a3 \\ a4 & a5 & a6 & a7 \\ a8 & a9 & a10 & a11 \\ a12 & a13 & a14 & a15 \end{matrix} \rightarrow \begin{matrix} a0 & a1 & a2 & a3 \\ a5 & a6 & a7 & a4 \\ a10 & a11 & a8 & a9 \\ a15 & a12 & a13 & a14 \end{matrix}$$

- **MixColumns**: In this stage, we multipy the matrixand use it to change the location of every bit inside a column of data.This phase is omitted during the last phase.

$$\begin{matrix} b0 \\ b1 \\ b2 \\ b3 \end{matrix} \equiv \begin{matrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{matrix} \times \begin{matrix} c0 \\ c1 \\ c2 \\ c3 \end{matrix}$$

- **Add Round Keys**: In this phase, the preceding stage's result is XOR'd with the matching round code. The sixteen bytes are interpreted like 2^7 bits of information instead of a rectangular structure. After all of the phases, the outcome is 2^7 bits of information encrypted. This process is continued till everything has been encoded.

3.3. Quantum Cryptography

Quantum encryption was the initial recorded quantum transmission technique, created by Charles H. Bennett as well as Gilles Brassard around 1984 as an element of an IBM laboratory effort merging mechanics and data.[23] Particle polarisation plus the uncertainty principle of Heisenberg are two of the fundamental concepts of quantum transmission that underpin quantum encryption. The photon is a type of light component that is described as a single package of electrical energy. Photons are continually in movement, and in a state of totally void area, photons move at exactly the same velocity as light to all viewers.[24] Light may be polarised, so its orientation may be altered from any angle. According to Heisenberg's Uncertainty Principle, there's an elementary limitation to the accuracy that can be achieved when confident pairings of a particle's physical characteristics (complementary variables)

may be evaluated at once. This idea is critical in quantum encryption since the condition of a polarised photon is known only at its point of observation.[26] Using this in encryption can keep hackers from stealing information. Furthermore, photons may be polarised in specified paths, preventing hackers from reproducing any publicly available information, for instance a qubit. This topic was initially presented in 1982 by W. K. Wootters and W. H. Zurek in their work "A single quantum cannot be cloned."[27]

Quantum encryption allows two people to reach an understanding on an arrangement of information despite ever coming together personally. Notwithstanding their lack of physical interaction, the method assures that the private key is only communicated among individuals. The essential premise of this technique is to encode every bit of the hidden key within the polarisation status of just one photon. It is important to note that, owing to this condition, a particle can't be captured sans getting destroyed, prohibiting an intruder from determining its status. A photon's state may be portrayed using one of 4 representations, each with a binary number of 0 or 1, as shown in the image 20 beneath.

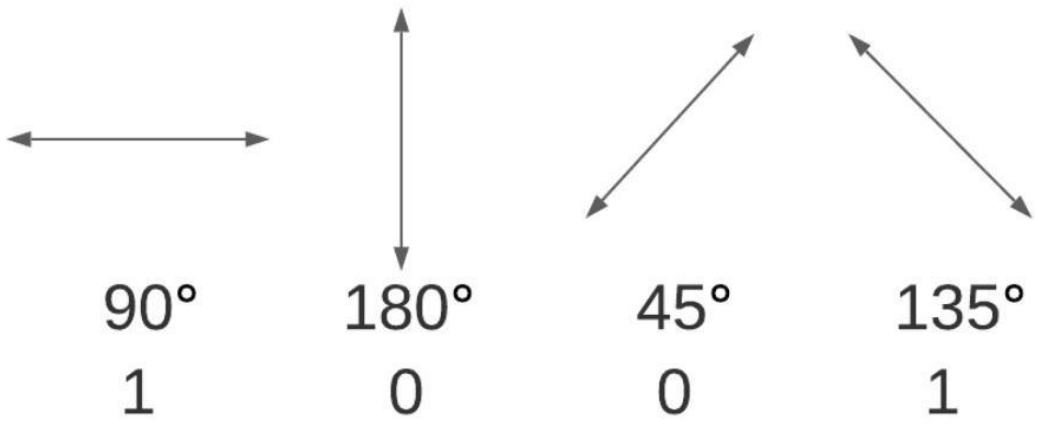


Figure 19. *Binary Values For The Filters Set By Sender*

Light passing through the filters marked with "-" and "|" is allowed, while light passing through filters marked with "/" and "\" may withstand the "X" filtering. Given such settings, the method has a transmitter C with a recipient

J. The protocol goes this way:

1. C gives a secret key as a bit string (e.g., 10010110) to J:
    a) C polarizes the light using filters "-", "|", "/", and "\" while J filters and measures the light using filters "+" and "X".
    b) C uses the filtering shown in image 20, and J selects between the "+" and "X" filters randomly to find the light angle.
2. J receives the bit string based on his and C's filters.
3. C and J compare their filters, and any bits where the filters do not match are deleted from bit phrase. The final byte form the last encryption key.

The last key is determined by the number of bits delivered, given that there has to be a fifty percent limit depending on the potential filter combinations. This protocol enhances security significantly because it is hard for an hacker to intercept the information. If an eavesdropper uses the incorrect key for a part, that part will be deleted. As depicted in the table below, if C and J employ the identical filtering, however the hacker selects the incorrect one, that bit will be lost, preventing the hacker from obtaining all the final bits. If the eavesdropper consistently uses incorrect filters beyond a certain threshold, C and J will detect the presence of an attacker and can restart the entire process.

***Table 3**. BB84 protocol*

Bit String	1	0	0	1	0	1	1	0
C's Filters	X	+	X	+	X	+	X	X
J' Filters	+	+	X	+	X	X	X	+
Eavesdropper Filter	X	+	+	+	X	X	+	X
Eavesdropper Filter		0	0	1	0		1	
Final Key with Eavesdropper Filter applied		0		1	0			

3.4. Specification For Designing The Model

This work presents a unique encryption technique for safely exchanging a secret key that uses combination of AES as well as a four-by-four Rubik's cube as the basis for encrypted and decoding. Although AES has been slightly modified for this method, the remaining phases stay same. The suggested strategy consists of two stages: the Automatic Evaluation System (AES) stage as well as the four by four Rubik's rotational keys stage.

3.4.1. Modified AES

Although much of the AES algorithm remains unchanged, modifications have been made to the shift rows step to accommodate the use of a four by four Rubik's Cube. Typically, during the shifting of the lines step, the rows of the four by four collections are moved 0 to 3, based on the line, as illustrated in image 21.

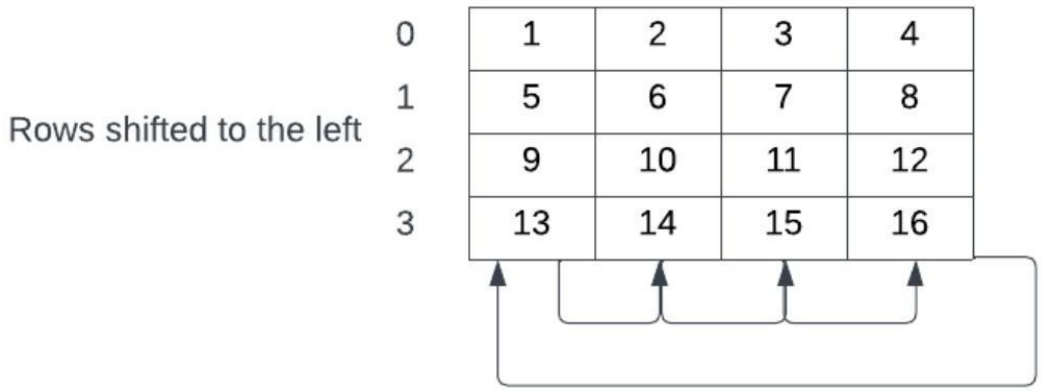


Figure 20. *The shift of AES*

During the changed AES model, the changed lines operation is adjusted based on the lengths of the provided plain-text and key. If both the plain-text and key lengths are 24 or 32 bytes, the collection is transformed into a six by six or eight by four collection. Each of the rows moved on the left have been modified to be between two and five with twenty-four bytes or between four and seven with thirty-two bytes, as shown in Diagram 21. This layout change was designed to allow a four-by-four Rubik's , as it includes 6 four by four arrays . This change improves safety by preventing a hacker from determining the total amount of the delivered encrypted text while the shifting of lines method is dependent upon the initial unencrypted text as well as key sizes. Once translated to a hex the encrypted text size increases to 192 digits. The Rotational Cypher Key is formed by combining the encrypted text with the previously generated rotational key utilising a XOR algorithm.

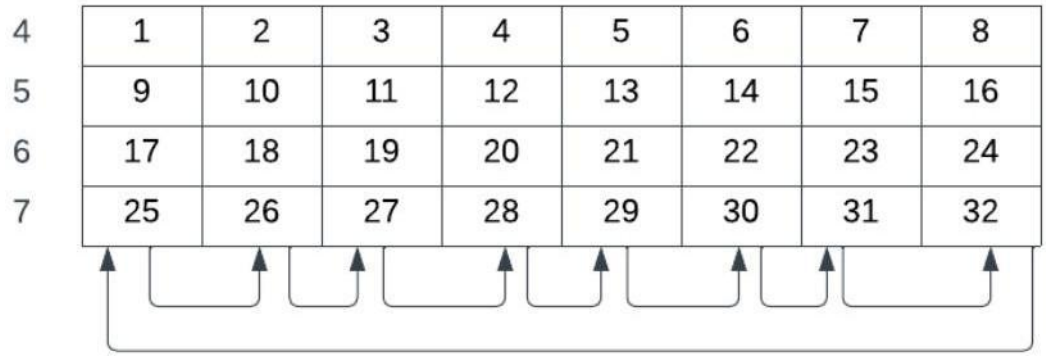


Figure 21. *Our Shift*

3.4.2. Rotation Key of Rubik's Cube

Upon creating the encrypted text with the revised AES technique, a rotating key may be generated. The corresponding rotation rule uses conventional Rubik puzzle notes and combinations. The image below shows a representation of a rotating key.

DA269IBC212

Figure 22. *Keyfor Rotation Phase*

The fragment shown represents the cube's rate of motion as well as the initial number of the the ciphertext The "DA2" section is read as follows:

- D: This denotes the bottom layer of the Rubik’s cube and indicates its movement direction, either clockwise or counterclockwise.
- A: Represents the direction of the movement, with "A" for anticlockwise and "C" for clockwise.

- 2: Indicates the number of rotations required, with possible values of 1 or 2, specifying the set number of rotations to be performed.

The integer "69" indicates what cube side that the initial encrypted text letter is given to. The rotary key comprises 96 motions, each chosen among a set of 72 potential cubic motions. To get the overall amount of feasible motion options, we compute 7296, producing roughly 2.01335175 x 10178. The chance of a hacker creating the same rotational key is around a one in 20 octoquinquagintillion. The following illustration shows a comprehensive instance of a rotating key.

```
FWA191IBC265UC235BC253IDC223RA280IRA174BWC116
IBC257UA159IFA124IDA226FA233UC275RWA132DA154UA185
ILC238FC182DC172IRC236UA151RA220DWA245ILA215RC29DC268
IBC211RWA23DWC221FWC188RA227UWA239DWC260IRC263RWC149
UA187RWA292DWA250IDC147RWC267FA25DC183ILC193IFC218
IFC177RA295IUA181FC231ILC189LWC28LWC230RWC184IBC186
ILA242FC162IDC14DWA270IBC190DC179IUA11BA261UC246
IDA256IRA143FA114ILC117DWA241IDC128IFA234LC110
UWC276FWC119DWC164LC171BWC212ILC269RC178DA255
BC173LA20IRA137DC122IFC129UWA125DC158IUC248
UWA113IUC16ILC140IDC194UC17IFA22RC152IUA144FWA266
```

Figure 23. *Full Rotation Key*

The rotary key is independent from the Rubik's puzzle colours, as soon as its locations have been established. A total of 96! potential choices for structuring the encrypted text depending on the method's cube motions. Furthermore, the changing key produced is completely randomised with every run of it. Lacking a rotated key, an intruder is unable to read the initial key or the reverse.

3.5. Implementing

Python has been chosen for the language of programming behind this method, which executes employing VSCode on a Raspberry Pi 4B running the operating system known as Raspberry Pi. The goal of this method is to provide greater safety than prior Rubik's puzzle methods and regular encryption methods.

3.5.1. Encryption

This step starts with the use of AES encryption, when it's completed, the encrypted text is allocated to each of the cube's sides. The figure illustrates the way the encryption text is set on a square before it is scrambled.

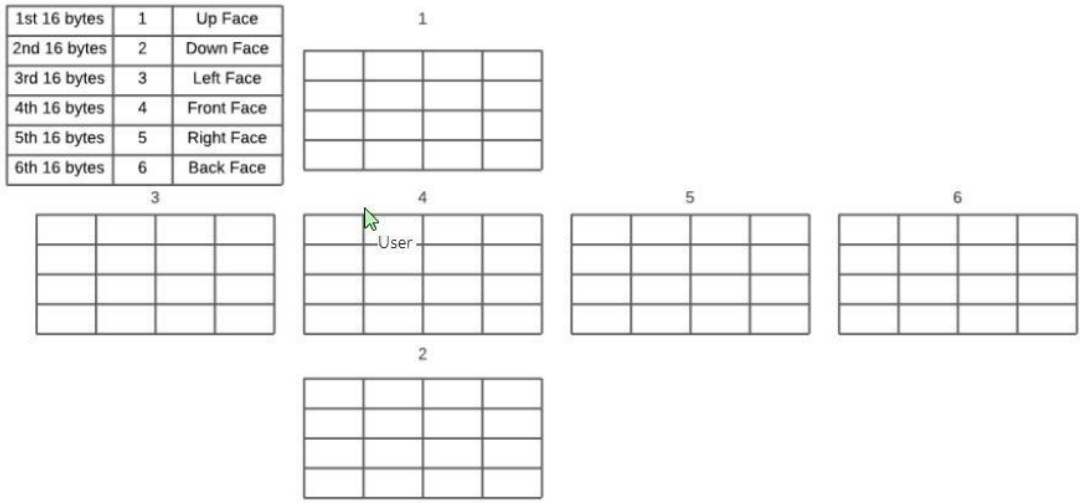


Figure 24. *Assigning the cypertext*

Once the rectangle is rotated, the encrypted text is modified to show the rotation, while a rotating key is generated to record the operation. The key in question could be similar to that seen in the illustration 24. The Rotating Key as well as the initial key can subsequently be XORed to form the Rotating Cypher Key, shown in the illustration 25.

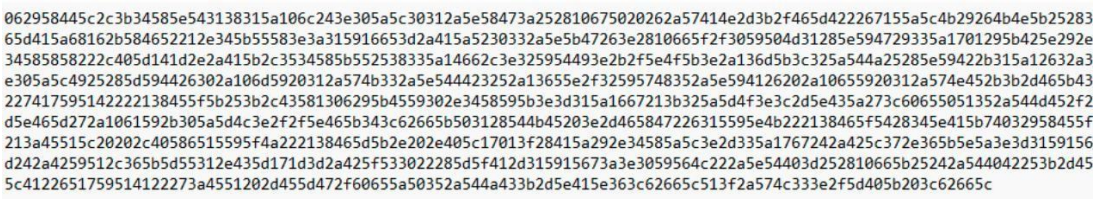


Figure 25. Cypher Key

3.5.2. Decryption

To decode the Rotating Cypher Key to return to plain text, you'll require the Rotary Cypher Key, the initial key, along with the jumbled encrypted text. Initially the primary key is translated to hex before being XORed onto the Rotational Cypher Key. The outcome is then translated again into hex to get the Rotational Key. Utilising this Rotational Key, every one of the encrypted text point is accurately allocated to the corresponding cube side depending on the number of them. The algorithm has to identify the right side, columns, and rows to ensure proper text placement.

For instance, if the encrypted text's location was forty-three, its index is determined by splitting forty-three by sixteen, yielding 2.68, that is converted to Two via the technique math.floor. To determine the columns, deduct sixteen Times two) off the encrypted text location integer forty-three, which equals eleven. Applying this number, the modulus of the method determines the value of the column, which yields Three (11 percent 4 equals 3). The row's amount is calculated by reducing Eleven with Four and then using the math.floor function once more, yielding a result of two. The image below shows the last side, columns, and rows structure: 2, 3, 2.

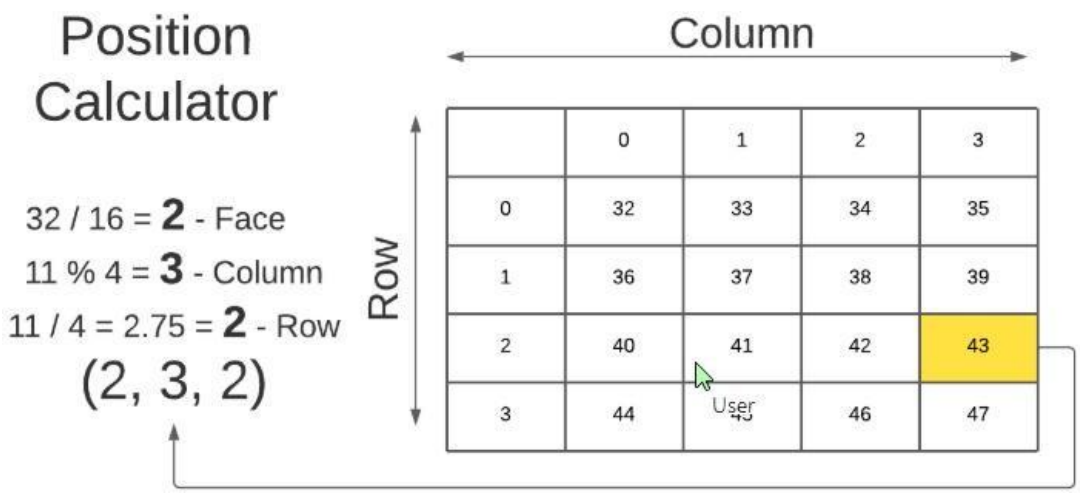


Figure 26. *Position Calculated for Rotation Key*

Utilising the above rotation key as well as appropriately placed encrypted text standards, the rectangular shape is unscrambled by undoing the motions shown in the rotary key, exposing the initial encrypted text without additional algorithmic alterations. For example, if the cube's initial motion is denoted as DA2, the inverse motion is AC2, where "A" meaning a one hundred and eighty in the opposite direction spin with "C" representing a one hundred and eighty clockwise motion. Decoding the encrypted text with the stages as well as keys described in the encryption key schedule will expose the initial unencrypted text.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1. Evaluation

In this section, three algorithms underwent testing on the Raspberry Pi 4, employing psutil to monitor CPU usage during the execution of encryption and decryption functions. Additionally, the time required for these operations was recorded.

4.1.1. CPU Utilization and Time Analysis for Each Algorithm

The evaluation included 3 methods: the suggested method, the encryption algorithm AES a hundred and twenty eight, as well as Ascon 128. "AES" [28] and "Ascon" [29] were developed entirely in Python, mimicking the technique of the suggested algorithm, to assure precision in calculating both encoding and decryption times. Each method went through ten runs to find the mean CPU utilisation and duration. The outcomes from these cycles are shown throughout Figure 27. Although AES and Ascon are limited to one hundred and eighty bits, the suggested method's assessment covers 192 along with 256-bit variations to demonstrate the distinctions between the different forms.

```
Proposed ALGORITHM

128 BITS
CPU USAGE ON ENCRYPT AND DECRYPT FUNCTIONS RUNNING - 49.2%
TIME TO ENCRYPT AND DECRYPT - 0.11026811599731445 seconds

192 BITS
CPU USAGE ON ENCRYPT AND DECRYPT FUNCTIONS RUNNING - 52.9%
TIME TO ENCRYPT AND DECRYPT - 0.1381371021270752 seconds

256 BITS
CPU USAGE ON ENCRYPT AND DECRYPT FUNCTIONS RUNNING - 56.9%
TIME TO ENCRYPT AND DECRYPT - 0.16170952796936035 seconds

------------------------------------------------------------

AES 128 (Python Implementation)
CPU USAGE ON ENCRYPT AND DECRYPT FUNCTIONS RUNNING - 16.4 %
TIME TO ENCRYPT AND DECRYPT - 0.0137837837838 seconds

------------------------------------------------------------

Ascon 128 (Python Implementation)
CPU USAGE ON ENCRYPT AND DECRYPT - 13.5 %
TIME TO ENCRYPT AND DECRYPT - 0.0022792816162109375 seconds

------------------------------------------------------------
```

Figure 27. *CPU Utilization and Time Analysis for Each Algorithm*

For the output the text ThisIsASampleTextForEncryptionTesting In the following table we show the results of some test runs. The table shows the key length used time for encryption and decryption for both AES and our proposed algorithm. It shows the mean results for multiple runs for each key size for both algorithms.

Table 4. *Table of Results*

Key Size	Algorithm	Mode	CPU	Time for Encryption	Time for Decryption
128 bit	AES	CBC	15.3%	0.00713 s	0.00812 s
128 bit	Proposed Algorithm	-	48.9%	0.0682 s	0.0565 s
192 bit	AES	CTR	20.3%	0.0106 s	0.00982 s
192 bit	Proposed Algorithm	-	55.7%	0.0825 s	0.06135 s
256 bit	AES	GCM	21.1%	0.0302 s	0.0209 s
256 bit	Proposed Algorithm	-	59%	0.1035 s	0.9403 s

4.2. Discussion

The primary objective of this study was to introduce a novel cryptography algorithm capable of delivering security while operating efficiently on IoT devices without significantly taxing their resources. Through the evaluation of these algorithms, it is evident that Ascon 128 outperforms AES 128, encrypting and decrypting data approximately 6.5 times quicker, while AES 128 is approximately 8.5 times quicker then the suggested technique. Evidently, the suggested technique has the poorest efficiency when it comes to of encrypting, decryption, and CPU utilization per execution. Based on the provided results, if speed and efficiency were the primary focus of this investigation, methods other than the proposed one, particularly Ascon, would be preferable. However, given that the research aims to prioritize both security and speed, It's worth noting that the proposed method ensures protection against brute force attacks, safeguarding the original plaintext and key. To establish the superior security of this algorithm compared to AES, further extensive testing would be necessary to assess its resilience. Given the considerable length of the final ciphertext compared to other algorithms, coupled with the unknown key to potential attackers, brute-forcing it would be a time-consuming endeavor.

CHAPTER 5

CONCLUSIONS

5.1. Conclusions And Future Research

The basic goal of this inquiry was to propose a cryptography algorithm that could offer security levels comparable to AES while operating efficiently on small IoT devices with limited resources.

This research initiative was driven by the considerations outlined in section 1 of the paper. By incorporating AES, a widely acknowledged industry-standard algorithm endorsed by NIST, into the proposed algorithm with slight modifications, the aim was to achieve this goal.

The proposed algorithm generates cipher-text using the modified AES algorithm, which is then encoded onto a four by four Rubik's puzzle, jumbled to generate an additional key. This approach was envisaged to enhance security levels beyond those of standard AES implementations, albeit with the trade-off of slightly longer encryption and decryption times, approximately 8% longer.

Given the study's results, this suggested method has an opportunity to improve safety for internet of things gadgets. However, significant efforts are required to enhance its performance while preserving its current level of security.

Future endeavours in this research domain would entail a comprehensive redesign of the algorithm to ensure adherence to best coding practices, as the current version is not yet optimized.

Subsequently, after enhancing the algorithm's performance, further benchmarking against other algorithms would be necessary to ascertain comparable resource usage on the test device. The suggested approach was tested on the Raspberry Pi four. It is imperative to assess its compatibility and effectiveness on various other IoT devices before contemplating real-world implementation.

REFERENCES

[1] Cso.ie. 2022. Information Society Statistics Enterprises 2021 - CSO - Central Statistics Office. [online] Available at: [Accessed 21 April 2024].

[2] Cso.ie. 2022. Internet of Things - CSO - Central Statistics Office. [online] Available at: [Accessed 21 April 2024].

[3] S. Adamović, I. Branović, D. Živković, V. Tomašević and M. Milosavljević, "Teaching interactive cryptography: the case for CrypTool," in *IEEE Conference ICEST*, 2011.

[4] Jovanovic, Philipp. (2015). Analysis and Design of Symmetric Cryptographic Algorithms.

[5] cryptography. Merriam-Webster.com. [Accessed 21 April 2024]

[6] Taylor, O. and Emmah, V., 2018. [ebook] Comparative Analysis of Cryptographic Algorithms in Securing Data: ResearchGate, p.122. Available at: [Accessed 30 April 2024].

[7] 2015. Biclique cryptanalysis of MIBS-80 and PRESENT-80block ciphers. [ebook] Wiley Online Library, pp.32, 33. Available at: [Accessed 30 April 2024].

[8] Van der Vieren, D., 2010. The Rubik's Crypto-Cube: a Trans-Composite Cipher. [ebook] Regis University, pp.16, 17. Available at: [Accessed 3 May 2024].

[9] Cubelelo. 2022. God's Number Explained: How Only 20 Moves Proved Enough to Solve Any Rubik's Cube Position. [online] Available at: [Accessed 3 May 2024].

[10] A. Kerckhoffs. La Cryptographie Militaire. *Journal des Sciences Militaires*, **IX**, 1883.

[11] E. Biham and A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems.

Advances in Cryptology — CRYPTO 1990. Volume **537**, Lecture Notes in Computer

Science. Springer, 1991

[12] D. Coppersmith. The Data Encryption Standard (DES) and its Strength Against

Attacks. *IBM Journal of Research and Development*, **38**(3), 1994.

[13] J. Guo, P. Karpman, I. Nikolić, L. Wang, and S. Wu. Analysis of BLAKE2. *Topics in Cryptology — CT-RSA 2014*. Volume **8366**, Lecture Notes in Computer Science. Springer, 2014.

[14] L. R. Knudsen and M. J. B. Robshaw. The Block Cipher Companion. Springer, 2011.

[15] J. Daemen, R. Govaerts, and J. Vandewalle. Correlation Matrices. *Fast Software*

Encryption — FSE 1994. Volume **1008**, Lecture Notes in Computer Science.

Springer, 1995.

[16] A. A. Selçuk. On Probability of Success in Linear and Differential Cryptanalysis. *Journal of Cryptology*, **21**(1), 2008.

[17] A. Bogdanov and E. Tischhauser. On the Wrong Key Randomisation and Key

Equivalence Hypotheses in Matsui's Algorithm 2. *Fast Software Encryption — FSE 2013*. Volume **8424**, Lecture Notes in Computer Science. Springer, 2014.

[18] J. Kaliski B. S. and M. Robshaw. Linear Cryptanalysis Using Multiple Approximations. *Advances in Cryptology — CRYPTO 1994*. Volume **839**, Lecture Notes in Computer Science. Springer, 1994.

[19] L. R. Knudsen and J. E. Mathiassen. A Chosen-Plaintext Linear Attack on DES. *Fast Software Encryption — FSE 2001*. Volume **1978**, Lecture Notes in Computer Science. Springer, 2001.

[20] N. T. Courtois, K. Nohl, and S. O'Neil. Algebraic Attacks on the Crypto-1 Stream Cipher in MiFare Classic and Oyster Cards. Cryptology ePrint Archive, Report 2008/166. 2008. ttp://eprint.iacr.org/2008/166.

[21] D. Khovratovich and I. Nikolić. Rotational Cryptanalysis of ARX. *Fast Software Encryption — FSE 2010*. Volume **6147**, Lecture Notes in Computer Science. Springer, 2010.

[22] Fortinet. 2022. What is the CIA Triad and Why is it important? | Fortinet. [online] Available at: [Accessed 12 May 2024].

[23] Shin, H., Kyoung Lee, H., Cha, H., Weon Heo, S. and Kim, H., 2019. IoT Security Issues and Light Weight Block Cipher. [ebook] Seoul: Hongik University, pp.382, 383. Available at: [Accessed 13 May 2024].

[24] 2022. Quantum Cryptography: An Emerging Technology in Network Security. [ebook] California State University, Northridge: Loyola Marymount University, p.15. Available at: [Accessed 14 April 2024].

[25] Jones, A., 2018. What Exactly Is a Photon?. [online] ThoughtCo. Available at: [Accessed 25 May 2024].

[26] 2015. Origin of Heisenberg's Uncertainty Principle. [ebook] Jaipur, India: NIMS University, p.203. Available at: [Accessed 20 May 2024].

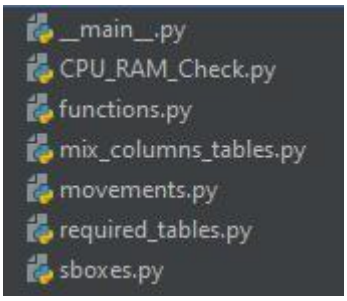
[27] Wootters, W. and Zurek, W., 1982. A single quantum cannot be cloned. [ebook] Williamstown, Massachusetts: Department of Physics and Astronomy, Williams College. Available at: [Accessed 22 May 2024].

[28] bozhu, 2012. AES-Python | Github [online] Available at: [Accessed 5 April 2024].

[29] meichlseder, 2014. Python implementation of Ascon | Github [online] Available at: [Accessed 5 April 2024].

APPENDIX

Some code for this thesis will me given in this section.



This is the organization for the code where the main file runs the code. CPU_RAM_Check contains the necessary libraries for checking the recourse utilization by the program for encrypting and decrypting.

The function file contains the functions for encrypting and decrypting. The matrices for encryption and decryption are given on the mix_columns_tables file. The movements file performs all the possible cube movements, it also has the movements array that stores a string value based on the function that was ran. The required_tables hold the indexes for each length of key.