National College of Ireland

Secure Cryptography Algorithm using Rubik's Cube for IOT Devices

MSc Research Project
Cyber Security

Kevin Kehoe
Student ID: X20147228

School of Computing
National College of Ireland

Supervisor: Ross Spelman

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Kevin Kehoe
Student ID:	X20147228
Programme:	Cyber Security
Year:	2022
Module:	MSc Research Project
Supervisor:	Ross Spelman
Submission Due Date:	15/08/2022
Project Title:	Secure Cryptography Algorithm using Rubik's Cube for IOT Devices
Word Count:	6075
Page Count:	19

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	*Kevin Kehoe*
Date:	13/08/2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	✓□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	✓□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	✓□

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Secure Cryptography Algorithm using Rubik's Cube for IOT Devices

Kevin Kehoe
x20147228

Abstract

The importance of the internet has increased exponentially during the COVID pandemic with more businesses and individuals moving their data to the cloud. Whenever we need data to be sent to another person or business, we rely on the internet to send this data. It's important that when sending sensitive data through the internet, it cannot be intercepted or stolen against unwanted individuals. Cryptography can allow businesses or individuals to transmit their sensitive data across the internet so that no one except the intended party can read the data. For any Cryptography algorithm to encrypt and decrypt sensitive data, mathematical equations are performed to alter the data. Depending on the complexity of these equations, more of the device's resources are used. In the IoT landscape, it is difficult to secure the data because of the computing power needed. Another problem with current Cryptography standards is Quantum Computing. It has been suggested that Quantum Computing will be 100 million times faster than any computer currently. Therefore, new algorithms need to be created with the constraints of IoT devices but still provide the security that strong algorithms such as AES provide. To tackle this problem a new Cryptography algorithm will be proposed while testing its security and resource management on an IoT device versus the top algorithms used today.

1 Introduction

For data to be sent throughout the internet, there are multiple ways for the data to be handled. It is important that the method chosen to transfer this data is secure enough to prevent unintended parties from reading the data but also reliable to prevent a failure in the data transfer. One of these methods is Cryptography which has become more important in providing security for many different types of applications especially since the COVID pandemic. The CSO's Information Society Statistics Enterprises 2021 report showed that "almost one in five (19%) enterprises reported an increase in sales via websites or apps during the COVID-19 pandemic" [1] while "one in ten set up a website to facilitate online sales" [1]. On the Internet of Things section of this report, nearly a quarter of these enterprises would use these devices to provide security to their premises. "A higher proportion of Manufacturing (26%) and Services (24%) enterprises use IoT devices for security than Construction (12%) firms" [2]. With many different enterprises utilising IoT device there is a bigger need to implement strong Cryptography algorithms.

When trying to implement a cryptography algorithm specifically in an IoT device there are several key challenges that need to be addressed:

- RAM and ROM on the device can be very limiting.
- The computing power is more restricted compared to standard PC's

- How fast the device can respond to a given command known as real-time.

While there are a huge variety of IoT devices, they are quite small and attempting to work with the limited resources can be quite the challenge. Some of these resources such as the ROM and RAM are needed to store information in memory and run an application, while trying to process the data can be restricted due to the low computing power. With these limited resources, trying to apply standard Cryptography standards to these devices could be deemed as unacceptable since the response needs to be quick and accurate while satisfying the needs for security. With IoT device usage increasing since the COVID pandemic, it is important that Lightweight Cryptography algorithms are designed to work on these devices. Lightweight Cryptography has the benefit in that its usage is not just limited to IoT devices, but it can be utilised in other types of devices that would not have the limited resources IoT devices are confined to such as PC's, servers, and mobile phones.

While many new algorithms have been proposed for both standard and IoT devices, classical Cryptography could potentially be affected by Quantum Computing. Since Quantum computers can do computations that classical computers cannot, they threaten the security of any secure communication. On top of this, quantum computers can break cryptographic keys that are generated because they are able to search all possible secret keys faster than any classical computer. This would allow an attacker or eavesdropper to intercept a communication between two parties. Shor's and Grovers algorithms can be implemented by quantum computers to break algorithms such as AES and RSA. With this threat towards classical Cryptography, new algorithms will need to be created to prevent Quantum computing from threatening the future of the field.

While there are many proposed Cryptography algorithms, a lot of them are tailored towards powerful systems and not IoT devices. Therefore, the research question of this paper is: Due to the computing power required, can a strong and powerful algorithm be created specifically for IoT devices?

This paper will investigate related work in Section 2 with the literature review focusing on the different types of cryptographic algorithms, a Rubik's cube algorithm that was slightly similar to the proposed algorithm for this paper and researching security concerns towards IoT. Section 3 will go into detail about how the AES algorithm encrypts data and how Quantum Cryptography works. Section 4 will show the Design Specification for the proposed algorithm and some slight changes made to the implementation of AES to better suit the Rubik's cube implementation. Section 5 will demonstrate the Implementation of the proposed algorithm for both encryption and decryption. Section 6 is the Evaluation phase which shows the case study that was done on the proposed algorithm and other algorithms on an IoT device, and section 7 will give a conclusion of the research carried out including the scope for future work on the algorithm.

2 Related Work

The Literature Review will represent recent studies made in different cryptography algorithms as well as security concerns for IoT devices. Sub section 2.1 will give a background on different cryptography algorithms, mainly based around their differences. Subsection 2.2 will discuss a cryptographic algorithm based around a 3x3 Rubik's cube. With the proposed algorithm being based on a 4x4 Rubik's cube, it is important to highlight the

similarities and what the proposed algorithm will do differently. Sub section 2.3 will address the security concerns towards IoT devices.

2.1 Different Types of Cryptographic Algorithms

In Cryptography, there are multiple methods for encrypting data. The first type of encryption is known as a symmetric key encryption which is when the algorithm produces a key that is used to both encrypt and decrypt data. The keys would be used to keep information private between two different parties. One of the drawbacks to this type of encryption is that both parties need access to the secret key, otherwise the data cannot be encrypted or decrypted unlike asymmetric encryption. The second type of encryption known as asymmetric encryption is when a pair of keys are used for encryption and decryption with one of these keys being public while the other is private. The public key would be used for the encryption phase while the private key would be used for the decryption phase. The final type of encryption is a hash function which is used to encrypt plaintext data of any length and turn it into unique ciphertext of a specific length. One of the strengths of a hash function is that since it is a flexible one-way algorithm it is nearly impossible to brute force all the possible combinations leading to the hash value.

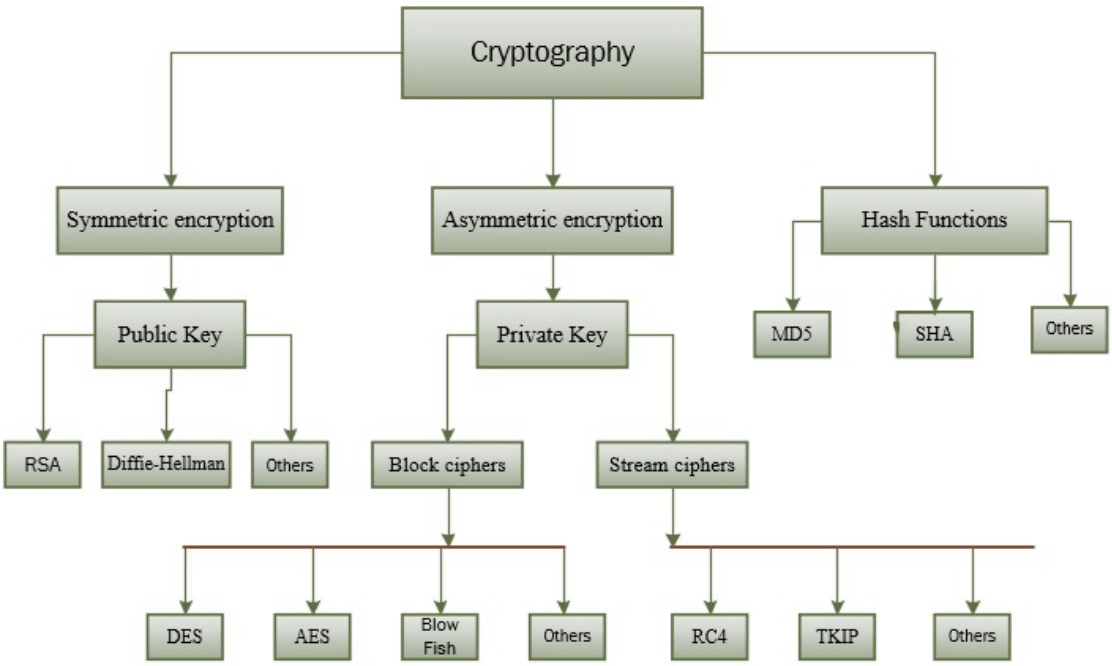

Figure 1 – Different types of encryption methods [3]

On the lightweight cryptographic side of encryption there is the Elliptic Curve Cryptography (ECC). ECC has been describe as the alternative technique to the RSA algorithm because it focuses on using pairs of public and private keys for both encryption and decryption. It can provide the necessary security by using mathematic elliptic curves. An Elliptic curve is like a normal curve, but it is over a finite field and uses the equation in Figure 2 to satisfy the equation.

$$y^2 = x^3 + ax + b$$

Figure 2 – Elliptic Curve Equation

It's important to note that with this curve, when adding two points together, the line created will only ever intersect one other point on this curve. The original point of intersection on this curve is known as the private key.

Comparative Analysis has been performed on both normal and lightweight algorithms in terms of security. One of the most important parameters of an encryption algorithm is the performance of the algorithm. The security levels of AES, DES and the RSA algorithms were tested with the results showing that the AES algorithm provided the highest level of security because it can use a variety of key lengths. Multiple attacks were attempted to crack the algorithm "like Square attack, Key attack, and Differential attack were tried, but none of them cracked AES algorithm" [4]. DES however had an inadequate level of security mainly because of the 56-bit length keys. "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" [4]. DES has been known for having weak S-boxes and with the short key lengths the algorithm is susceptible to an attack. The security of the RSA algorithm is based on how difficult it can be to factorise a large integer. The public key consists of two numbers, one being the multiplication of the two prime numbers. If this prime number would be factorised the private key would be compromised. This process would be very time consuming and because of this, RSA is proven to be a strong algorithm. Between these three algorithms, ranking them from most to least secure, AES would be the most secure with DES being the least secure.

On the lightweight cryptography side, algorithms such as PRESENT and SIMON are well known in the field to be suitable for IoT devices in terms of encryption and decryption. PRESENT can function with a 64-bit block and its key size can be either 80 or 128 bits. Researchers have proven that from biclique cryptanalysis on the 80-bit version, "uses 527 S-boxes in aa full-round encryption so the computational complexity is as follows:" [5]

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

[5]

SIMON was introduced in June of 2013 and is optimised for performance in hardware implementations. "The Simon block cipher with 2n-bit block and mn-bit key is denoted Simon 2n/mn where n must be 16, 24, 32, 48, or 64 and m must be 2, 3 or 4. For example, Simon32/64 refers to the version of Simon acting on 32-bit plain text blocks and using a 64-bit key" [6]. From the comparative analysis on these algorithms, it was stated that these two algorithms had great performance in terms of tech, power, and area usage. While these algorithms are suitable for IoT encryption and decryption, attacks are still possible as shown in figure 4.

AES is the most powerful and secure algorithm based on the comparative analysis, but it is important to highlight the level of security both PRESENT and SIMON provide for the low computational power needed and how they are suitable for IoT.

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, Boomerang, 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, good for RFID tagging.	Impossible differential attack on 26th round, Biclique cryptanalysis
[23]	CLEFIA	128	128	18	FN	0.13	2.48	2488	39	Has fast encryption and decryption, lesser rounds, energy efficient	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

Figure 4 [7]

2.2 Rubik's Crypto-Cube – Trans-Composite Cipher

On May 9th, 2010, a paper was released discussing the possibility of a cryptography algorithm based on a 3x3 Rubik's cube. It took the weaknesses of the Vigenère and Mitchell algorithms to create a new algorithm that prevents attacks from anagramming and frequency analysis.

The Vigenère Cipher is known as a polyalphabetic shift cipher because for all plaintext characters, they can point to multiple ciphertext letters based on either a keyword or a specific shift. If our plaintext would be THISISATEST and our key would be JUMP, our key would be extended until its length is the same as the plaintext. Our ciphertext would result in the following ciphertext based on Figure 5

T	H	I	S	I	S	A	T	E	S	T
J	U	M	P	J	U	M	P	J	U	M
C	B	U	H	R	M	M	I	N	M	F

Figure 5 – Vigenère Cipher Encryption

Mathematically, to get our ciphertext our plaintext and keywords should be converted into vectors that have numbers which are mapped to a letter of the alphabet. With the keyword JUMP, if this were to be changed into a vector of numbers, if the numbers 0 – 25 are mapped to the alphabet in the same order (A = 0, B = 1 ….) The keyword would be converted to the

following vector: (9, 20, 12, 15). If the same is applied to the first four characters of the plaintext the vector would be (19, 7, 8, 18). In Figure 5, if using the plaintext T and the key J gives the ciphertext of C then using the converted vector values of 9 and 19 would give us the ciphertext vector value of 2. This is calculated by using the modulo calculator which is that the formula a mod b = c. The modulo operator gives us the remainder of the value of a divided by b so 28 mod 25 (0 – 25 indexes for alphabet a- z) gives the remainder of 3 which based on the index value for the alphabet gives the letter C.

On a normal 3x3 Rubik's cube, there are 54 pieces 6 sides and nine cubies each. Mitchell's system involves some editing of the plaintext before the shuffling begins. "The first step in the process requires one to write the numeral "1" on the upper left square of a cube face. The number "2" can be on an arbitrarily chosen square on another face, and so on until all six sides have a characteristic numeral as its representative identifier" [8]. Starting with the face with the number 1 written on it, any empty cubies is filled with the plaintext character and this process then goes to the face with the number 2 until all the cubies on the cube are filled. An example of this is shown in figure 6 with the plaintext value given to be: LEARNINGCRYPTOGRAPHYINNCIHASBEENAGREATEXPERIENCE

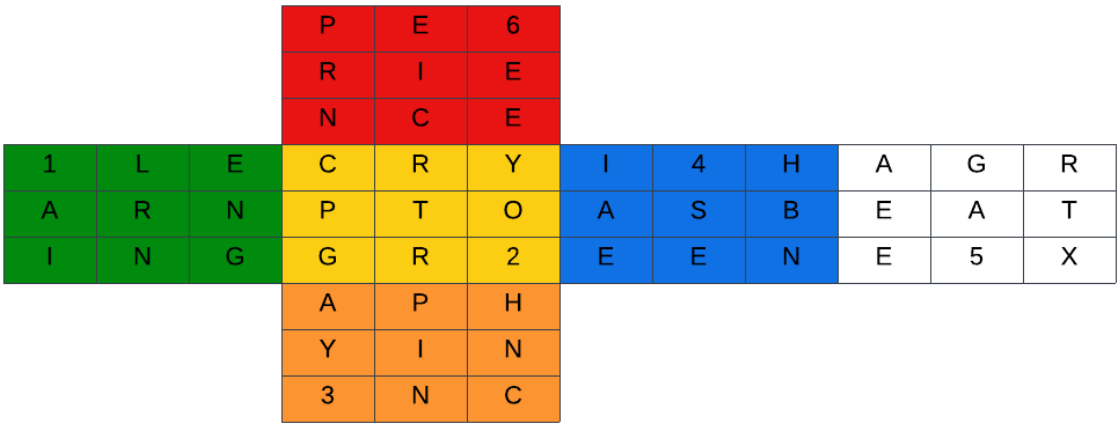

Figure 6 – Mitchells Algorithm – First step

The second step in this algorithm is to generate the rotation key. This key will allow the cube to rotate back into what was setup in step one. The number 1 must be the starting sequence in the top left corner of whatever face is chosen. With this point mentioned, the "rotation key has more than 7.25 billion different encryptions due to the ways the plaintext can be initialized. Successful attacks cannot occur because of a lack of frequently repeated plaintext passages. Multiple anagramming as an attack would also be unreliable because this attack relies on the repeated use of the same ordering of the transposition" [8]. Another step taken by Mitchell was to propose an order of assigning the faces of the cube based on a six-letter extension for example "ACEDBF". This special order would let the receiving party know how to assign the ciphertext to the cube. To scramble the cube, the requirements are that R equals to Row, C equals to Column and L equals to Level. 1, 2 ,3 would represent the clockwise rotation in multiples of 90-degree angles and 4, 5, 6 would represent the outside 2 layers of the cube being rotated in multiples of 90-degree angles. With those requirements set, an example of a rotation key would be as shown in Figure 7.

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

Figure 7 – Example of Mitchells Rotation Key

One of the problems of cryptography regarding a Rubik's cube is there are bound to be instances where the same characters are repeated in the ciphertext which can decrease the actual size of the key. "Studies and research have proven that God's Number is number 20" [17]. Gods number for a 3x3 Rubik's cubes is the maximum number of moves that would be needed to solve any of the "43 quintillions or to put it in perspective, 43,252,003,274,489,856,000 permutations combinations of the cube" [17].

2.3 Security concerns regarding IoT

Security is of upmost importance to achieve a private and safe way of communication between individuals, software's, and other parties. There are three different processes that make up the required triad.

1. Availability: To ensure that two different individuals or parties want to send and receive data, there needs to be a way to identify and authenticate both. The big issue with this in the world of IoT is that outside of its users, there can be other services involved with the device in use. To ensure safe authentication between different devices or parties there needs to be a procedure set before any interaction between the two are required. "These can be programmed to become available when the primary system has been disrupted or broken" [18].
2. Integrity: When transferring data between two individuals or parties it is of upmost importance that the data is not tampered with between its travel. The accuracy of the data being sent needs to be one hundred percent and any lower should be of concern to its integrity. With IoT devices exchanging data between multiple devices its accuracy can be affected due to the low computing power. IoT should follow traditional security procedures but currently it is not being following due to limited resources.
3. Confidentiality: Even if the integrity of the data isn't affected and there is a standard procedure between the two different parties, if the data can be revealed by an unauthorised user, the data would be deemed as public rather than private. To control access to private information for a party, that party must have an authorisation system to determine who needs to have access to the necessary privileges. With IoT users not being limited to just human users but also to different services, it is important that during its life cycle, the data is protected to ensure confidentiality.

While CIA triad is important towards security, in IoT there are two other processes that should be a requirement which are Energy Efficiency and Heterogeneity. Since "each device of IoT operates on battery power or self-harvested energy sources, so it is critical to reduce energy consumption when each device functions" [9]. If the energy is not reduced, it can shorten the lifecycle of the device in use which can influence the IoT's network. With IoT devices already able to incorporate heterogeneity devices, there can be a wide variety of devices connected to a network. Security measures need to be implemented to handle all types of heterogeneity devices.

In the field of Cryptography, there needs to be an energy efficient algorithm and many lightweight block ciphers were proposed during 2013 to the National Security Agency. The main two algorithms proposed were SIMON and SPECK but “they were criticized by many security researchers and have been rejected from ISO standardization in 2015” [9].

IoT can be broken down into three different layers which are as shown in Figure 8.

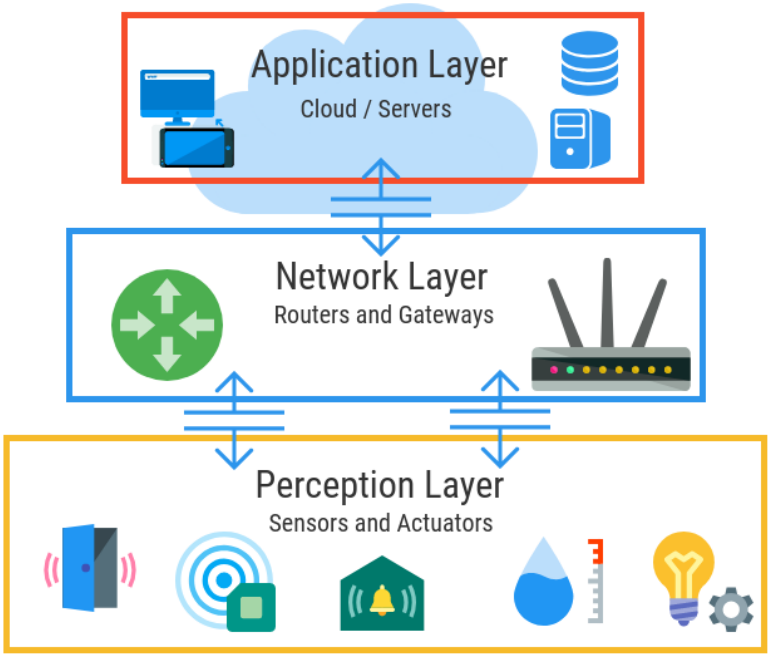

Figure 8 – Three Layers of IoT Architecture [10]

Unfortunately, with each of these layers there are security issues which can potentially affect cryptographic data.

2.3.1 Perception Layer

This layer can consist of smart devices which can contain sensors or microcontrollers. The high processing power of these devices can have the capability to connect with the network which can connect to the application running. Since this layer is responsible for collecting data, it’s important that this data is authenticated from the device before being sent to the network layer. The two security issues with this layer are tampering with the nodes and injection malicious code. Tampering with the nodes can allow an attacker to gain access to confidential data such as the keys to a cryptographic algorithm. If software is not updated, it could allow the attacker to inject code which would allow them to gain access to the network.

2.3.2 Network Layer

Data created from the perception layer, is transmitted across the network which makes up the Network layer. There are more possible threats from this layer than the perception layer because data can be transmitted from multiple heterogeneous devices. Some of the possible attacks that can be performed on this layer are routing, DDoS, and man-in-middle attacks. Routing attacks changed the path that the data is being sent to, which can affect the integrity

of the data. Most devices today are known to be heterogeneous, so any other malicious device can create a DDoS attack which could affect the network. Man-in-Middle attacks can occur if the devices that are sending and re receiving the data are not secured.

2.3.3 Application Layer

This layer receives the data from the Network layer and uses that data to provide the necessary function of the IoT infrastructure. “There are several different challenges at the application layer. For example, data access permissions and identity authentication can be a cause for concern. With all the different types of applications and users, it is difficult to manage access permissions and authentication” [11]. Some threats at this level that can occur are data leakage and misconfiguration of the device. Data leakage can be a security risk “if the attacker knows the vulnerabilities of the application. This can cause the manipulating of data that is stored on cloud” [12]. If an IoT device is not configured correctly in terms of security like an operating system or a database, the device could be open to attacks causing damage to the whole IoT network.

3 Methodology

From the literature review uncovering security issues regarding all layers of IoT, new cryptographic algorithms need to be designed to fight against quantum computing. Due to the low computing power of IoT devices, providing the adequate security is challenging. The proposed algorithm will improve on the generation of the Rubik’s Crypto-Cube rotation key while mixing in the AES algorithm to create a powerful cipher. This section will discuss the AES algorithm and an overview on Quantum Cryptography to highlight the power and speed compared to current cryptography algorithms.

3.1 AES algorithm

The AES algorithm also known as Advanced Encryption Standard is a block cipher which was sent to the National Institute of Standard and Technology in the year 2000. It was created to replace the DES algorithm since there were security vulnerabilities found with it. It is currently known as the most secure algorithm compared to other algorithms. For it to encrypt and decrypt data being sent, the data is split into a fixed size of 128 bits (16 bytes) and works based off a matrix. The key sizes for AES can be one of three sizes which are 128, 192 or 256 bits. Depending on the size of this key, the algorithm is performing actions either 10, 12 or 14 rounds. Figure 9 will illustrate the actions taken for each round.

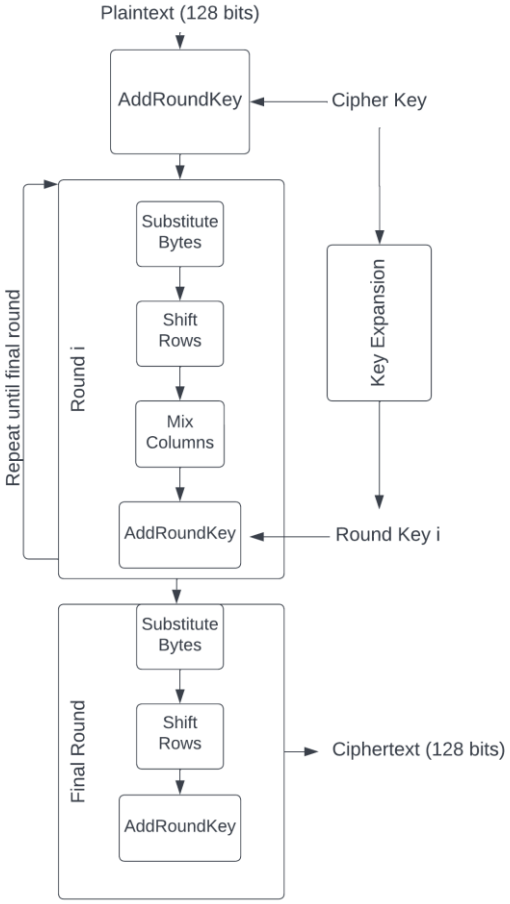

Figure 9 – AES Encryption Architecture

3.2 Quantum Cryptography

Quantum cryptography is the first known quantum distribution scheme and was "developed by Charles H. Bennett and Gilles Brassard in 1984 (BB84) as part of research study between physics and information at IBM lab" [13]. Quantum Cryptography requires two parts of quantum distribution which are the photos polarisation and the Heisenberg Uncertainty principle. "A photon is a particle of light defined as a discrete bundle (or quantum) of electromagnetic (or light) energy. Photons are always in motion and, in a vacuum (a completely empty space), have a constant speed of light to all observers" [14]. Since light can be polarised, its direction can be reflected to any angle that is desired. "Heisenberg's uncertainty principle states that there is a fundamental limit to the precision with which certain pairs of physical properties of a particle (complementary variables) can be measured simultaneously" [15]. This principle plays a crucial role in quantum cryptography because the light particle or photon being polarised will only ever be known from the moment that the point is measured. If this is applied to cryptography, it can prevent any eavesdropper from sniffing the data. Also, photons can be polarised in a specific direction thanks to the photon polarisation principle which again can also stop an eavesdropper from copying any known data such as a qubit. This was first mentioned in 1982 when W. K. Wootters and W. H. Zurek published their paper on "A single quantum cannot be cloned" [16].

Quantum cryptography works by two parties agreeing on a sequence of bits but without them ever meeting face to face. While they can't meet in this way, the BB84 protocol can ensure that its secret key is exclusively shared between just them. The theory behind this protocol is to alter all a secret key's bits into the polarization state of a single photon. It's important to note that because of this state, a photon cannot be measured since measuring it would destroy it, which prevents an eavesdropper from reading its state. The state of a photon can be denoted by one of 4 symbols which each could carry either a binary value of 0 or 1 as shown in Figure 10 below.

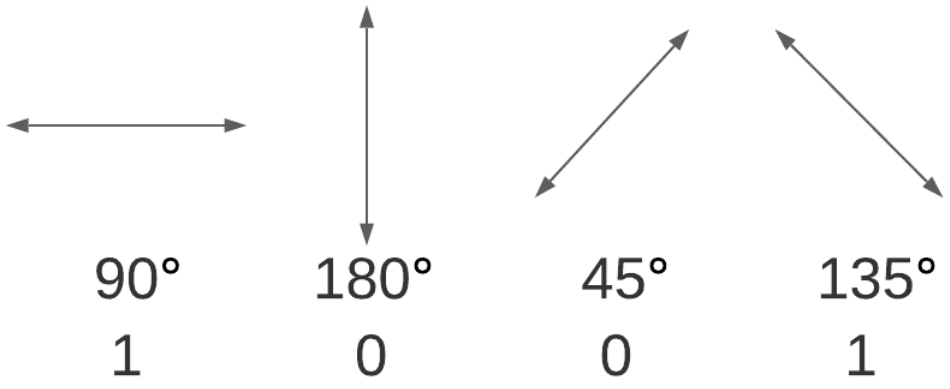

Figure 10 – Filters being set from the sender with possible binary values

Light – and | are allowed to pass through the filter + whereas / and \ light can pass through the filter X. With these parameters the BB84 protocol works with having a sender (Carly) and a receiver (John). The steps taken with this protocol would be:

1. Carly would send a secret key as a bit string (10010110) to John
    a. Carly would use the – | / \ filters to polarise the light while John would use the + and X filters to filter and measure the light being passed.
    b. Carly would use the filters set in Figure 10 while John would randomly select the two filters + and X to measure the angle of the light.
2. John would receive the bit string based on his filters sent from Carly as well as the filters used by Carly.
3. The filters sent from both Carly and John are compared and where the filters do not match, that bit would be removed from the string bit as shown in Figure 11. The remaining bits are used as the final key for the encryption.

This final key will also have been established by how many bits were sent as there needs to be a 50% threshold based on the possibilities of the filters. The security with this protocol is heightened since its impossible for an eavesdropper to sniff the data because if they use the wrong filter for a bit, that bit would be destroyed. Figure 11 shows that if Carly and John use the same filter but the eavesdropper used the wrong filter, that bit would be destroyed, and the eavesdropper would not get all the bits of the final key. If the wrong filters used by the eavesdropper exceed a threshold, Carly and John will know that an attacker is there and can restart the whole process.

Bit String	1	0	0	1	0	1	1	0
Carly's Filters	X	+	X	+	X	+	X	X
John's Filters	+	+	X	+	X	X	X	+
Eavesdropper Filters	X	+	+	+	X	X	+	X
Final Key		0	0	1	0		1	
Final Key with Eavesdropper filters applied		0		1	0			

Figure 11 – BB84 protocol example

4 Design Specification

This paper will propose a new cryptographic algorithm to ensure secure sharing of a cryptographic key based on AES and a 4x4 Rubik's cube for both encryption and decryption AES has been slightly modified in this algorithm, but the implementation of the other steps is the same. There are two phases for the proposed algorithm which are the AES phase and the 4x4 Rubik's rotation key.

4.1 AES Modified

While most of the AES algorithm is the same, the shift rows have been altered since this algorithm will be used with a 4x4 Rubik's Cube. Usually on the shift rows step the 4x4 matrix rows are shifted from 0 to 3 depending on the row as shown in Figure 12.

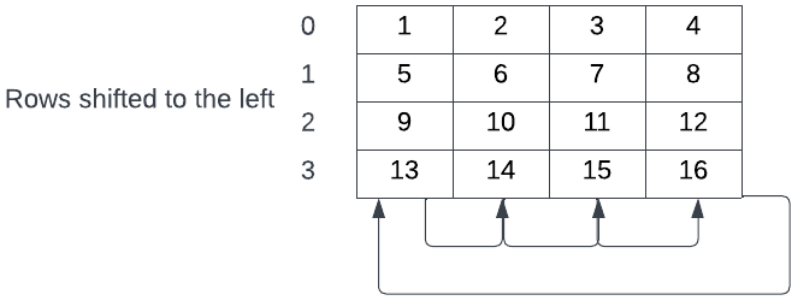

Figure 12 – Normal AES Shift Rows

In the modified version of AES, the shift rows depend on the length of the plaintext and key provided. If the length of both is either 24 for 32 bytes in length the matrix is changed to be either a 6x4 or 8x4 matrix. The rows shifted to the left are changed to be either 2 – 5 for 24 bytes or 4 - 7 for 32 bytes as shown in Figure 13.

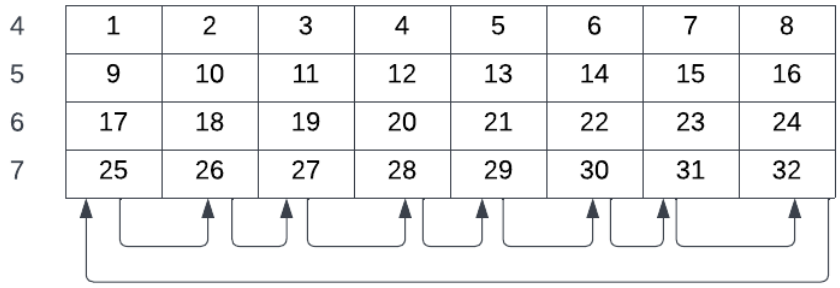

Figure 13 – Modified AES Shift Rows

This change in design was to accommodate a 4x4 Rubik's cube since there are six 4x4 matrixes (96 cube faces). This can add an extra level of security since an attacker will not be able to determine the length of the ciphertext returned since the Shift Rows operation depends on the plaintext and key originally sent. The ciphertext length is extended to be 192 characters in length when it is hexed. The ciphertext and the rotation key created are joined together using an XOR operation to create the Rotation Cipher Key. When decrypting the ciphertext into plaintext only the first N characters will be kept. This N value depends on the length of the original key.

4.2 Rubik's cube Rotation Key

After the ciphertext has been generated with the modified AES algorithm, the rotation key can be created. The rotation key is designed with official Rubik's notations and positionings in mind. A snippet of a rotation key is shown in Figure 14.

```
DA269IBC212
```

Figure 14 – Rotation Key Snippet

The snippet DA269 represents the cubes movement and the position of the first value of the ciphertext. The DA2 is broken down into the following:

- D – this is the notation for the Rubik's cube, this will move the bottom layer of the cube either clockwise or anticlockwise.
- A – This can be A for anticlockwise or C for clockwise.
- 2 – This value will be either 1 or 2 and indicates in sets of rotations how many rotations need to be done.

The value of 69 indicates which cube face the first ciphertext is assigned to. The rotation key is generated from 96 movements with each movement being 1 of 72 possible cube movements. To calculate all the possibilities of movements that can be chosen, we must calculate 72^96 which gives us the value 2.01335175E178. The chances of an attacker getting the exact same rotation key is around twenty octoquinquagintillion. A full example of a rotation is key is shown in Figure 15

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

Figure 15 – Full sample of Rotation Key

This rotation key does not require any knowledge of the colours of the Rubik's cube if it has its positions. There are 96! possible combinations for the ciphertext to be arranged based on the cube movements chosen by the algorithm. Finally, the rotation key generated is completely random from each execution. Without the rotation key, the attacker would never be able to decipher the original key and vice versa.

5 Implementation

For this algorithm, Python was the programming language chosen and is run using VSCode on the Raspberry Pi 4B using the Raspberry Pi OS. The goal of this algorithm is to provide greater security than previous implementations of Rubik's cube algorithms and classic cryptography algorithms.

5.1 Encryption Phase

This phase starts with AES encryption and after encrypting, the ciphertext is assigned to the faces on the cube. After assigning the ciphertext onto the cube, Figure 16 will demonstrate how the text is assigned before the cube is scrambled.

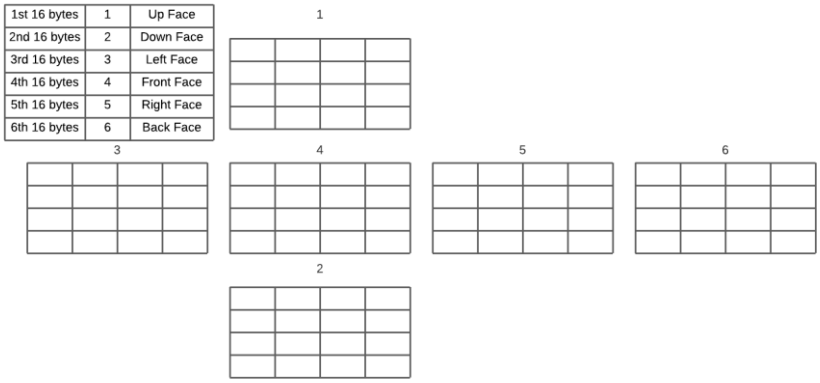

Figure 16 – How ciphertext is assigned to 4x4 cube

After this, the cube is scrambled, the ciphertext is changed to reflect the scramble and a rotation key is generated to tell the cube how it was scrambled. This key can look like the key

in Figure 15. Both the Rotation Key and the original key are XORed together to generate the Rotation Cipher Key which can look as shown in Figure 17.

```
062958445c2c3b34585e543138315a106c243e305a5c30312a5e58473a252810675020262a57414e2d3b2f465d422267155a5c4b29264b4e5b25283
65d415a68162b584652212e345b55583e3a315916653d2a415a5230332a5e5b47263e2810665f2f3059504d31285e594729335a1701295b425e292e
34585858222c405d141d2e2a415b2c3534585b552538335a14662c3e325954493e2b2f5e4f5b3e2a136d5b3c325a544a25285e59422b315a12632a3
e305a5c4925285d594426302a106d5920312a574b332a5e544423252a13655e2f32595748352a5e594126202a10655920312a574e452b3b2d465b43
22741759514222213845 5f5b253b2c43581306295b4559302e3458595b3e3d315a1667213b325a5d4f3e3c2d5e435a273c60655051352a544d452f2
d5e465d272a1061592b305a5d4c3e2f2f5e465b343c62665b503128544b45203e2d465847226315595e4b222138465f5428345e415b74032958455f
213a45515c20202c40586515595f4a222138465d5b2e202e405c17013f28415a292e34585a5c3e2d335a1767242a425c372e365b5e5a3e3d3159156
d242a4259512c365b5d55312e435d171d3d2a425f533022285d5f412d315915673a3e3059564c222a5e54403d252810665b25242a544042253b2d45
5c4122651759514122273a4551202d455d472f60655a50352a544a433b2d5e415e363c62665c513f2a574c333e2f5d405b203c62665c
```

Figure 17 – Example of Rotation Cipher Key

5.2 Decryption Phase

To decrypt the Rotation Cipher Key back into plaintext, we need to provide the Rotation Cipher Key, the original key and the ciphertext after the cube had been scrambled. The original key is hexed and XORed with the Rotation Cipher Key and then the result is de-hexed to get the Rotation Key. With this rotation key each position of the ciphertext can be assigned to the correct cube face based off its number. To position the text correctly the code must know which face, column and row the text must be assigned on.

If the position of the ciphertext is 43 for example, the index calculated will be 43 / 16 which is 2.68 which turns into 2 since the math.floor method is used. To calculate the column, the ciphertext position value 43 needs to be misused from (16 * 2) which gives us 11. Using this value, the column can be calculated by using the modulus equation which would result in the number 3 – (11 % 4 = 3). The row value is calculated by dividing 11 into 4 and by using the math.floor method again like before the value will be 2. The final values in the face, column row arrangement will be 2, 3, 2 as shown in Figure 18.

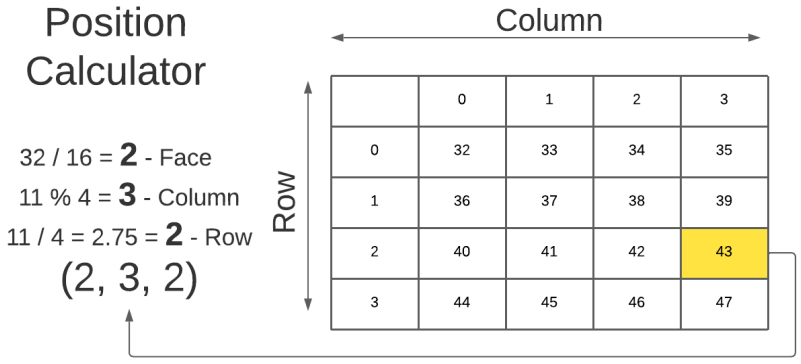

Figure 18 – Position Calculated for Rotation Key

With this rotation key and the ciphertext values being positioned correctly, the cube is descrambled by checking the opposite movement of the rotation key to show the original ciphertext before any of the modified AES algorithm steps are applied. If the cube movement were to be DA2 as explained in section 4.2, the opposite movement would be DC2 since the A would be anticlockwise 180 degrees whereas C would be clockwise 180 degrees. Decrypting the ciphertext from the steps and keys used in the key schedule will uncover the original plaintext.

6 Evaluation

In this section, three algorithms were tested on the Raspberry Pi 4 using psutil to check for the CPU usage when the encrypting and decrypting functions would be executed. The time taken to perform this was also measured.

6.1 CPU Usage and Time for each algorithm / Case Study 1

The three algorithms that were tested were the proposed algorithm, AES 128 and Ascon-128. "AES" [19] and "Ascon" [20] were pure Python implementations like the proposed algorithm to get a more accurate time to complete the encryption and decryption phases. Each algorithm was executed ten times each to get an average CPU usage and time. Figure 19 will show the results of the algorithms. While AES and Ascon are limited to 128 bits, the proposed algorithm will show results for 192 and 256 bits also to show the differences between the different versions.

```
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 19 – CPU and time results for each algorithm

6.2 Discussion

The main aim of this research was to propose a new cryptography algorithm that could be secure but run on IoT devices without using too much of the device's resources. From testing these algorithms, we can prove that Ascon 128 encrypts and decrypts data around six and a half times faster than AES 128 whereas AES128 runs around eight and a half times faster than the proposed algorithm. Evidently the proposed algorithm is the slowest in terms of encryption and decryption as well as the CPU usage per execution. Considering the results that have been provided, if the primary focus of this research were to be speed and efficiency of the algorithm then the methods other than the proposed one be considered mainly Ascon. However, since the focus of this research is to provide security and speed, a brute force attack for example would be unable to get the original plaintext and key since the algorithm would be secure. To prove that this algorithm is more secure than AES, more tests would need to be

ran against the algorithm to ensure its security strength. With the final ciphertext being so long compared to the other algorithms and having the key unknown to an attacker, it would take a long time for it to be brute forced.

7 Conclusion and Future Work

The main goal behind this research was to propose a cryptography algorithm that could provide at least the security of AES while being able to run on a small IoT device without using too many resources available to it. The motivation to conduct this research was based off section 1 of this paper. With AES having been proven to be an industry standard algorithm by the NIST, this goal would be ensured since AES was implemented into the algorithm while being slightly modified. The proposed algorithm would generate the ciphertext with the modified AES algorithm before being placed on a 4x4 Rubik's cube and scrambled to produce another key. This would provide a higher level of security compared to the standard AES implementation with the sacrifice being that it took 8% longer to encrypt and decrypt.

Considering the results of the research, there is a possibility that this proposed algorithm could be implemented for security towards IoT devices, but a lot of work would need to be carried out to improve performance while maintaining its current security level. The future scope for this research would be to redesign the algorithm from the ground up to ensure that the best coding practises are used since the algorithm is not yet optimised. Secondly, after improving this algorithms performance, more benchmarking needs to be carried out against other algorithms to ensure that the resources being used on the test device is the same as the other algorithms. While the proposed algorithm was tested on a Raspberry Pi 4, there are other IoT devices that this algorithm needs to be tested on before it could possibly be implemented for real world use.

Acknowledgement

I would like to personally thank my supervisor Ross Spelman for his constant support, guidance, and encouragement throughout the duration of this research dissertation. I would also like to thank my girlfriend, family and all my peers enrolled in the Cyber Security Masters programme who helped in guiding me through any difficult times I had throughout the year. I want to thank the National College of Ireland for giving me the opportunity to work on my research and for providing me all of the resources needed to complete it.

References

[1] Cso.ie. 2022. Information Society Statistics Enterprises 2021 - CSO - Central Statistics Office. [online] Available at: <https://www.cso.ie/en/releasesandpublications/ep/p-isse/informationsocietystatisticsenterprises2021/> [Accessed 28 June 2022].

[2] Cso.ie. 2022. Internet of Things - CSO - Central Statistics Office. [online] Available at: <https://www.cso.ie/en/releasesandpublications/ep/p-isse/informationsocietystatisticsenterprises2021/internetofthings/> [Accessed 28 June 2022].

[3] Alrammahi, M. and Kaur, H., 2014. Development of Advanced Encryption Standard (AES) Cryptography Algorithm for Wi-Fi Security Protocol. [image] Available at: <https://www.researchgate.net/publication/323369289_Development_of_Advanced_Encrypti

on_Standard_AES_Cryptography_Algorithm_for_Wi-Fi_Security_Protocol> [Accessed 30 June 2022].

[4] Taylor, O. and Emmah, V., 2018. [ebook] Comparative Analysis of Cryptographic Algorithms in Securing Data: ResearchGate, p.122. Available at: <https://www.researchgate.net/publication/333755102_Comparative_Analysis_of_Cryptographic_Algorithms_in_Securing_Data> [Accessed 30 June 2022].

[5] 2015. Biclique cryptanalysis of MIBS-80 and PRESENT-80block ciphers. [ebook] Wiley Online Library, pp.32, 33. Available at: <https://onlinelibrary.wiley.com/doi/epdf/10.1002/sec.1375> [Accessed 31 June 2022].

[6] 2014. A Hardware Implementation of Simon Cryptography Algorithm. [ebook] Ferdos Blvd: Islamic Azad University Science and Research Branch, p.246. Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6993386> [Accessed 2 July 2022].

[7] 2017. A Review on Lightweight Cryptography Algorithms for Data Security and Authentication in IoTs. [image] Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8269731> [Accessed 2 July 2022].

[8] Van der Vieren, D., 2010. The Rubik's Crypto-Cube: a Trans-Composite Cipher. [ebook] Regis University, pp.16, 17. Available at: <https://epublications.regis.edu/cgi/viewcontent.cgi?article=1511&context=theses> [Accessed 4 July 2022].

[9] 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: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8669029> [Accessed 4 July 2022].

[10] Calihman, A., 2022. Architectures in the IoT Civilization. [online] NetBurner. Available at: <https://www.netburner.com/learn/architectural-frameworks-in-the-iot-civilization/> [Accessed 6 July 2022].

[11] 2018. Securing the Internet of Things (IoT): A Security Taxonomy for IoT. [ebook] Altoona-PA USA: Pennsylvania State University, p.164. Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8455902> [Accessed 10 July 2022].

[12] 2022. Evaluating Security Threats for each Layers of IoT System. [ebook] Lahore: University of Engineering and Technology, p.4. Available at: <https://www.researchgate.net/publication/336149742_Evaluating_Security_Threats_for_each_Layers_of_IoT_System> [Accessed 11 July 2022].

[13] 2022. Quantum Cryptography: An Emerging Technology in Network Security. [ebook] California State University, Northridge: Loyola Marymount University, p.15. Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6107841> [Accessed 14 July 2022].

[14] Jones, A., 2018. What Exactly Is a Photon?. [online] ThoughtCo. Available at: <https://www.thoughtco.com/what-is-a-photon-definition-and-properties-2699039> [Accessed 15 July 2022].

[15] 2015. Origin of Heisenberg's Uncertainty Principle. [ebook] Jaipur, India: NIMS University, p.203. Available at: <https://www.researchgate.net/publication/280042462_Origin_of_Heisenberg's_Uncertainty_Principle> [Accessed 20 July 2022].

[16] Wootters, W. and Zurek, W., 1982. A single quantum cannot be cloned. [ebook] Williamstown, Massachusetts: Department of Physics and Astronomy, Williams College. Available at: <https://www.nature.com/articles/299802a0> [Accessed 22 July 2022].

[17] Cubelelo. 2022. God's Number Explained: How Only 20 Moves Proved Enough to Solve Any Rubik's Cube Position. [online] Available at: <https://www.cubelelo.com/blogs/cubing/how-to-solve-rubiks-cube-in-20-moves#:~:text=The%20God's%20Number%20is%20the,30%20second%20using%20F2L%20method> [Accessed 2 August 2022].

[18] Fortinet. 2022. What is the CIA Triad and Why is it important? | Fortinet. [online] Available at: <https://www.fortinet.com/resources/cyberglossary/cia-triad#:~:text=The%20three%20letters%20in%20%22CIA,and%20methods%20for%20creating%20solutions> [Accessed 2 August 2022].

[19] bozhu, 2012. AES-Python | Github [online] Available at: <https://github.com/bozhu/AES-Python> [Accessed 5 August 2022].

[20] meichlseder, 2014. Python implementation of Ascon | Github [online] Available at: <https://github.com/meichlseder/pyascon> [Accessed 5 August 2022].