

THE STUDY OF SYMMETRIC AND ASYMMETRIC

KEY ENCRYPTIONS

by

Johnathan Bevers

A thesis presented to the Honors College of Middle Tennessee State University

in partial fulfillment of

the requirements for graduation from the University Honors College

Fall 2021

Thesis Committee:

Dr. Yi Gu, Thesis Director

Dr. Jaishree Ranganathan, Second Reader

Dr. Rebekka King, Thesis Committee Chair

THE STUDY OF SYMMETRIC AND ASYMMETRIC
KEY ENCRYPTIONS

by

Johnathan Bevers

APPROVED:

Dr. Yi Gu, Thesis Director

Associate Professor of Computer Science

Dr. Jaishree Ranganathan, Second Reader

Assistant Professor of Computer Science

Dr. Rebekka King, Thesis Committee Chair

Associate Professor in the Department of Philosophy and
Religious Studies

ACKNOWLEDGEMENTS

This project would not have been possible without the support of many people. First and foremost, I am extremely grateful to my supervisor, Dr. Yi Gu for her invaluable advice, continuous support, and patience during my thesis study. She has also helped me extensively with formatting and proofreading many different revisions. I would also like to thank Dr. Jaishree Ranganathan for joining the committee as a second reader and Dr. Rebekka King for being on the committee, as well as their valuable and insightful comments.

I am very grateful to Middle Tennessee State University for awarding me the Honors Transfer Fellowship and providing me with the resources necessary to take on such a large project. This thesis has been the longest and hardest project I have done so far in my studies. I have learned many life skills and lessons that will stick with me as I venture into other research topics or education goals.

ABSTRACT

Encryption has become a mission critical component in the concept of cyber security. As we know, cryptography prior to the modern age was effectively synonymous with encryption, which has a large amount of complex information that needs to be understood for encryption algorithms to make sense for an average person. In this thesis, we present the background information needed to understand modern day encryption, including mathematical background, encryption keys, security level, and other relevant topics. We also introduce and discuss cryptography and encryption in fine detail to provide an assessable reading experience for those who are both professionals in the field and general readers. Moreover, we implement several classic encryption algorithms in Python, such as DES, AES, RSA, Diffie-Hellman and Elgamal encryption, by analyzing the differences and similarities among them. We evaluate and compare the performance of those algorithms based on a list of carefully selected performance metrics, such as security and speed, to determine which algorithms should be used and which ones should be avoided in different scenarios. The study reveals that different encryption algorithms may have different performances in various scenarios, which can be used as a guidance when a security system is being developed and implemented.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....	I
ABSTRACT	II
TABLE OF CONTENTS.....	III
LIST OF FIGURES	V
LIST OF TABLES	VI
LIST OF TERMS.....	VII
CHAPTER 1: INTRODUCTION	1
CHAPTER 2: RELATED WORK.....	3
CHAPTER 3: MATHEMATICAL BACKGROUND.....	6
3.1 GROUPS.....	6
3.2 PRIME NUMBERS AND GCD.....	7
3.3 MODULAR ARITHMETIC	7
3.4 THEOREMS	8
3.5 COMPUTATIONAL AND UNCONDITIONAL SECURITY	9
CHAPTER 4: SYMMETRIC KEY ENCRYPTION.....	11
4.1 DES	13
4.2 AES	14
4.3 A FLAWED SYMMETRIC KEY ALGORITHM.....	16
CHAPTER 5: ASYMMETRIC KEY ENCRYPTION.....	18
5.1 RSA	20
5.2 DIFFIE–HELLMAN KEY EXCHANGE.....	23
5.3 ELGAMAL ENCRYPTION	23

5.4 ELLIPTIC-CURVE CRYPTOGRAPHY	24
5.5 TWO OF THE BEST-KNOWN USES OF PUBLIC KEY CRYPTOGRAPHY	25
5.5.1 <i>Digital Signature</i>	25
5.5.2 <i>Public Key Certification</i>	26
CHAPTER 6: PERFORMANCE ANALYSIS AND COMPARISON	29
6.1 AES VS DES	31
6.2 AES VS RSA	31
6.3 DIFFIE-HELLMAN VS ELGAMAL ENCRYPTION	32
6.4 RSA VS DES	33
6.5 SYMMETRIC KEY VS ASYMMETRIC KEY	33
CHAPTER 7: CONCLUSION	35
BIBLIOGRAPHY	37

LIST OF FIGURES

Figure 1. An Example of Rail Fence Cipher.....	12
Figure 2. Input State Array	15
Figure 3. Representation of digital signature.....	25
Figure 4. Public Key Certification	27

LIST OF TABLES

Table 1. Performance Comparison among Five Encryption Algorithms	29
Table 2. RSA with Varying Key Sizes	30

LIST OF TERMS

Plaintext Information that is ordinary readable and is the input for an encryption algorithm.

Ciphertext The output that is produced by passing plaintext into the encryption algorithm.

Decryption The conversion of ciphertext back to plaintext.

Encryption The conversion of plaintext into ciphertext.

Key Information that is used by an encryption algorithm to either encrypt or decrypt information.

Symmetric Key Encryption A type of encryption that involves a singular shared key for both encryption and decryption.

Asymmetric Key Encryption A type of encryption algorithm that involves using two separate keys for encryption and decryption. The two keys are known as a Public Key and a Private Key.

Cryptography The practice of securing information so that information is hidden to everyone other than the intended recipients.

Cryptanalysis The study of analyzing encryption systems to attempt to learn enough information to be able to gain access to encrypted contents.

Cryptosystem A group of 3 algorithms that consists of one for key generation, one for encryption, and one for decryption.

Brute Force Attack A form of attack on an encryption algorithm that is performed by attempting to guess every valid key until the correct one is found.

CHAPTER 1: INTRODUCTION

As both e-commerce and technology grow more complex, information is being passed through networks and systems at a record rate. Internet users send information between one another, with some which contain sensitive information such as passwords, credit cards, or anything that one deems private. Used in everyday tasks to protect information from being assessed by unauthorized users, cryptography is the study and practice of secure communication. Two phases, encryption and decryption lay out the basis for cryptography. Encryption is the conversion of plaintext into ciphertext where plaintext is information that is needing to be hidden from unauthorized users and ciphertext is the encrypted form of the plaintext that is illegible without being decrypted. Encryption algorithms are combined with plaintext and a key as input to form the ciphertext and sends it over to the intended recipient. Decryption is when that ciphertext is converted back into plaintext, making the text legible. Ciphertext should be useless to everyone except users that have the key and decryption algorithm. Ciphertext is an important part of cryptography to have a form to store information in that cannot be read by unauthorized individuals. There are two main forms of encryption. Symmetric key encryption and asymmetric key encryption. There are differences and similarities between these two types of encryptions that will be discussed.

In the beginning of my research, I had assumptions that asymmetric encryptions were always the better encryption algorithm to use due to the amount of security they can provide. As I continued to research, I discovered that the speed of symmetric key

encryption algorithms gives them a large benefit in a variety of situations, but speed is not always the most important attribute. My research is important because it allows the reader to establish the foundation needed to understand cryptography while at the same time showing real results of certain algorithms being tested. The results obtained in this thesis show the main similarities and differences among encryption algorithms and what makes them unique. The reader should finish reading with a better overall sense of encryption algorithms and the process from start to finish.

CHAPTER 2: RELATED WORK

“A Survey of Encryption Standards” written in 1993 by Burt Kaliski [3] is a good representation of how far the field of cryptography has come since the beginning. In the paper, Kaliski compared multiple encryption algorithms based on their properties and states multiple environments where the encryption algorithms were possibly useful. Simple systems such as secured emails were listed as potential uses. Most of the attention is focused on public-key cryptosystems, key-agreement algorithms, authentication codes, and secret-key cryptosystems. The study is mainly for experienced cryptographers in the field due to the large amount of jargon used and the year it was published. In 1993, computational technology was just now starting to reach everyday households and it was still in a constant process of discovery. Another paper titled “Encryption algorithms comparisons for wireless networked sensors” written by Xiaohua Luo *et al.* in 2004 shows the transforming field of cryptography, being applied to wireless networks [12]. During the time when the paper was written, resources surrounding security in wireless networks were limited and still in the works. The authors produced a survey of potential encryption algorithms that could be used along with wireless networks. The survey includes a performance analysis to determine the amount of memory required for each encryption algorithm presented and compares it to the resources currently available.

A step from the beginning to more recent studies can offer a drastic change in the types of research being done. More and more research efforts have been devoted into the limits of certain encryption algorithms, such as a paper titled “DNA Image Encryption Using Modified Symmetric Key (MSK)” written by Arti Ochani *et al.* [9]. In this paper, DNA is used to further secure an encryption algorithm and provide an additional security layer for a symmetric key encryption algorithm. Another example of current work in the field is a paper titled “A New Symmetric Key Encryption Algorithm using Images as Secret Keys”, written by Mazhar Islam *et al.* [2]. In this research paper, instead of the traditional keys being used for an encryption algorithm, an image is used as an input for an encryption algorithm to provide a unique way of encryption. Studies such as these two are constantly being performed and most studies require a significant amount of previous knowledge to understand. Encryption is becoming more popular and required as more of our information and sensitive materials are moved to an electrical means of storage. Increased computing power and resources has led to certain algorithms, such as DES, to be broken and replaced by future algorithms.

The main contribution of our work is to provide a survey of both symmetric and asymmetric key encryption algorithms and conduct performance analysis to determine the best environments for each encryption algorithm type. Our survey intends to be generalized enough for the general readers but specialized enough to contain enough information to be useful in the field of cryptography. Even with the increase in technology that we currently

have, there are still limitations when it comes to learning, implementing, and understanding encryption algorithms. We provide a necessary background to allow the audiences to understand the mathematical concepts behind them and then provide the performance analysis to show the algorithms in action with their results.

CHAPTER 3: MATHEMATICAL BACKGROUND

Modern day encryption algorithms are based on mathematical concepts to increase the security provided and the speed at which information can be processed. These mathematical foundations can be quite complicated to the ordinary people who do not study cryptography. Therefore, we present below a list of important mathematical concepts to facilitate our discussion on the encryption algorithms and minimize the unnecessary confusions.

3.1 Groups

The first complicated math topic that will be explained is a group. A group is a set of elements with a binary operation that is performed to each ordered pair of elements. The first kind of group we will talk about is called an abelian group. A group is abelian if the group's binary operation can be applied to two elements in the group and the result is not impacted by the order the two elements are in. The abelian group is sometimes referred to as the commutative group due to it expressing the commutative property. There's another type of group called a cyclic group. All cyclic groups are abelian, and they can be infinite or finite. A group is cyclic if every element of the group can be generated by an element contained within the group. This can be very confusing, so, to help explain further here is an example. We'll start with the set $U(9) = \{1,2,4,5,7,8\}$. This is an example of a cyclic

group because you can obtain each element within the group with the element 2^k . Such as $2^0 = 1, 2^1=2, 2^2=4, 2^3=8, 2^4 \bmod 9 =7, 2^5 \bmod 9 = 5$. This makes 2 the generator of the group $U(9)$ and means that it is a cyclic group.

3.2 Prime Numbers and GCD

Prime numbers play an important role in cryptography due to their usage in many encryption algorithms. A number is considered prime if it can only be divided evenly by 1 and itself. The number 1 is not prime due to itself and 1 being the same. 2 is the only even prime number because it is divisible by 1 and itself (2). Another topic that is useful for encryption is the greatest common divisor (GCD). The GCD of 2 or more numbers is the largest number that can divide into each of the numbers. The most common format for GCD is $gcd(i,j)$. For example, $gcd(9,12) = 3$, because 3 is the greatest common divisor for the integers 9 and 12. If the GCD of two numbers is 1, then those two numbers are considered relatively prime. Two numbers are considered relatively prime if their only positive common divisor is 1. For example, the integers 7 and 31 are relatively prime because 1 is their only positive common divisor.

3.3 Modular Arithmetic

The remainder when dividing numbers is called the modulo. For example, if you have the expression $7 \bmod 2 \equiv 1$ it would portray that if you divide 7 by 2, 1 would be the remainder. Modular arithmetic is a confusing topic that is sometimes hard to understand by non-cryptographers. The easiest way to comprehend this subject is through time. On an analog

clock there are only 12 numbers but there are 24 hours in a day. If it was 5:00 and you wanted to predict where the hour hand would be in 10 hours, it would not be at 15:00. 3:00 is the correct time it would be at. This is the same as saying $15 \bmod 12 \equiv 3 \bmod 12$. In this example, 15:00 and 3:00 are equal or in modular arithmetic terms, congruent. Congruency is commonly represented by an equal sign with 3 lines instead of the usual 2.

3.4 Theorems

In cryptography, there are a couple of number theory concepts called theorems that are commonly used in encryption and decryption. The first theorem is called Fermat's Little Theorem but is sometimes shortened to Fermat's theorem. Fermat's theorem is as stated: If p is a prime number and a and p are relatively prime then $a^{p-1} \equiv 1 \pmod{p}$. This theorem can be useful for calculating complex problems that would be difficult to solve otherwise. Here is an example of the theorem being used to solve a complex problem. If you have the question "Find the remainder when you divide $3^{100,000}$ by 35," it would be difficult to find the solution without this theorem. 31 is a prime number and 31 and 3 are coprime so the theorem can be used. Coprime is another term that is a synonym for relatively prime. Here are the steps:

$$3^{31-1} \equiv 1 \pmod{31}$$

$$3^{30} \equiv 1 \pmod{31}$$

This is what we get when we input the information into the theorem. 100,000 will be divided by 30 to get the quotient and remainder.

$$\frac{100,000}{30} \text{ quotient} = 3333 \text{ and remainder} = 10$$

We then raise both sides to the power of the quotient.

$$(3^{30})^{3333} \equiv (1)^{3333} \pmod{31}$$

$$3^{99,990} \equiv 1 \pmod{31}$$

$$(3^{10})3^{99,990} \equiv (3^{10}) \pmod{31}$$

After computation the power of 3 is 10 away from 100,000 so we add 3^{10} to both sides.

$$3^{100,000} \equiv 59049 \pmod{31}$$

$$3^{100,000} \equiv 25 \pmod{31}$$

This means that if $3^{100,000}$ was divided by 31 then 25 would be the remainder. This example shows the use of the theorem to simplify an otherwise complicated question.

The next theorem that will be discussed is known as Euler's theorem. Euler's theorem is the generalized version of Fermat's theorem. Euler's theorem states that

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

For example, if you have $a=3$ and $n=10$ then you can get $\phi(10)=4$ by counting all of the numbers that are coprime to 10 in modulo 10. Using these values, we can get $3^4 = 81 \equiv 1 \pmod{10}$. Euler's theorem is very important when it comes to the public key encryption algorithm, Rivest-Shamir-Adleman (RSA).

3.5 Computational and Unconditional Security

Computational security is a common term used to describe an encryption scheme when it is unbreakable with the current computing technology available or if the materials required

to break the scheme cost more than the value of the information encrypted. Another term used to describe encryption schemes is unconditionally secure. If an encryption is unconditionally secure, it means that the encryption algorithm is unbreakable by any method of attack. The only current algorithm that is unconditionally secure is the One Time Pad (OTP). One Time Pad was proposed by an officer in the Union Army by the name of Joseph Mauborgne in 1882 [10]. It is an improvement to the Vernam cipher that results in a very secure encryption scheme. A random key is used that is the same size as the message that is being encrypted. This key is used only once to encrypt and decrypt a message and then it is discarded. The ciphertext that is produced from the encryption has no relation to the original plaintext, so it is impossible to break. Now that we have explored an overview of mathematical concepts relevant to cryptography, we will pivot to a discussion of symmetric key encryption.

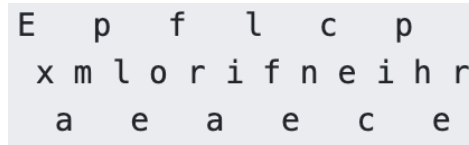
CHAPTER 4: SYMMETRIC KEY ENCRYPTION

Symmetric key encryption is one of the two major encryption algorithm types. In symmetric key encryption, a shared key is used for both encryption and decryption. There are many different types of symmetric key encryption algorithms, and they can range in performance. In this chapter we will be discussing a range of encryption algorithms such as DES, AES and a proposed encryption algorithm that contains a flaw.

One of the earliest types of symmetric key encryption is called a substitution encryption. A substitution encryption is when plaintext is swapped with other letters and symbols. An early form of the substitution method is one created by Julius Caesar, and it is called the Caesar Cipher. The Caesar Cipher is a very simple algorithm that is not very secure. The cipher involves shifting the plaintext letters forward or backwards in the alphabet depending on the shift. For example, with a shift of 3 the plaintext letter “A” would be encrypted with the ciphertext of “D” and the plain text letter “Z” would be encrypted as “C”. Mathematically the Caesar Cipher is expressed as $C=E(x) = (x + n) \text{ mod } 26$ where x is the letter you are wanting to encrypt, and n is the number of letters you want it to be off by. The easiest method to break the cipher is brute forcing due to the limited number of characters in the alphabet.

A second form of symmetric encryption is transposition encryptions. Transposition encryptions encrypt plaintext by performing permutations on it. An example of a basic

transposition cipher is a rail fence cipher. In the rail fence cipher, plaintext is written in diagonals. The number of rows of plaintext is the key which is also called rails for this cipher. The number of rails is the depth level. For example, to set up the plain text message “Example of rail fence cipher” with a key of 3 would look like (3 rails)



E	p	f	l	c	p						
x	m	l	o	r	i	f	n	e	i	h	r
a	e	a	e	c	e						

Figure 1. An Example of Rail Fence Cipher

The ciphertext would be “Epfclpxmlorifneihraeaece” Rail fence is a transposition cipher that is easy to crack due to the simplicity of the encryption. Pure transposition techniques are easily broken due to having the same letters as the plaintext. This can be countered by performing multiple permutations or by performing the same encryption multiple times. Most secure algorithms combine the use of transposition and substitution methods.

Symmetric key encryptions can either be classified as a block cipher algorithm or stream cipher. Stream ciphers encrypt data each bit at a time and block ciphers encrypt an entire block of plaintext at once to produce ciphertext that is the same length as the plaintext. In block ciphers the size of the plaintext is usually 64 or 128 bits. In both block and stream ciphers, a key is shared between two parties. Each method of encryption has its own usage; block ciphers are more commonly used in network related applications compared to stream

ciphers. Now that the basis of symmetric key encryptions has been discussed, the first symmetric key algorithm being studied in this thesis is called DES.

4.1 DES

DES is short for Data Encryption Standard. It was one of the most popular encryption schemes from 1977 until 2001. DES is an example of a block cipher because it uses a 56-bit key to encrypt a block of 64-bit plaintext. The input and output size are the same at 64 bits. DES is also based on a Feistel cipher, and it is the same as a Feistel cipher other than the beginning and ending permutations. All Feistel ciphers use the same methods for encryption and decryption. The encryption and decryption for DES uses the same algorithm but the subkeys, first permutation and third permutation are all reversed. The DES encryption process is broken down into 3 phases. The first phase consists of putting the 64-bit block of plaintext through a permutation to produce the permuted input. The second phase consists of 16 rounds of the same function involving substitutions and permutations. During the final round the output is 64 bits and is a function of the permuted input and key. The output of the 16th round is split into two sections, the first half and the second half. These halves are swapped and are run through the initial permutation used to produce the permuted input. The output of this procedure produces the complete ciphertext.

Now that the encryption algorithm has been explained, the security of DES will be discussed. DES was initially secure against brute force attacks, and it experiences the avalanche effect which makes it more secure to being cracked. The avalanche effect occurs

when changing the plaintext slightly has a large impact on the output of the ciphertext. This occurs in DES due to the drastic change of the text after each round it goes through. The text may not be changed much after the 1st round but after 16 rounds the ciphertext is considerably different than the original plaintext. The avalanche effect can also be caused by changing the key used. Since the key length in DES is 56 bits, there are only a possible 2^{56} number of different keys. During the time when DES was first created, this size of key was more than sufficient due to the computing power of computers at that time. In our current era, computers have multiple cores and can run multiple brute force attempts at once. Because of the increase in computing power, DES is not as secure as it once was. Contemporary encryption algorithms recommend using a key of at least 128 bits to be considered secure against brute force attacks. 128 bit is twice the size of the key used in DES. Eventually DES became too vulnerable to use in everyday applications so another encryption algorithm called Triple DES was developed. Triple DES is just applying the DES encryption algorithm 3 times to have a more secure encryption. Triple DES has not currently been broken but some theoretical attacks have been proposed.

4.2 AES

Eventually DES was replaced by strong algorithms that could compete with the current computing power. One of the symmetric algorithms to replace DES is called AES or also known as Advanced Encryption Standard. AES is a block cipher and takes in a plaintext block of 128 bits and uses a varying number of key sizes. 128, 192, or 256-bits can be used

as the key size. When a certain key size is used the bit size is attached to the name. For example, when AES is used with a 128-bit key then it is AES-128. These key sizes are larger when compared to DES. A larger key makes the encryption more secure. The amount of processing the encryption algorithm goes through depends on the key size. There are 10 rounds of encryption for 128-bit sized keys, 12 rounds for 192-bit, and 14 rounds for 256-bits. In each key size the rounds are the same except the last round in each size. Each round consists of a permutation step on the rows, a mixing step on the columns, a substitution step, and the addition of the key. These rounds are performed in a different order for encryption and decryption. The plaintext block is put into a 4 x 4 matrix. The plaintext bytes are inputted into the array in this order.

$$\begin{bmatrix} \text{byte}_0 & \text{byte}_4 & \text{byte}_8 & \text{byte}_{12} \\ \text{byte}_1 & \text{byte}_5 & \text{byte}_9 & \text{byte}_{13} \\ \text{byte}_2 & \text{byte}_6 & \text{byte}_{10} & \text{byte}_{14} \\ \text{byte}_3 & \text{byte}_7 & \text{byte}_{11} & \text{byte}_{15} \end{bmatrix}$$

Figure 2. Input State Array

This is known as the state array. Each row and column is also known as a word since they are both 4 bytes. The array created from the plaintext is called the input state array because it is being inputted into the encryption algorithm. The array produced after the processing rounds is called the output state array. AES is a variation of the Rijndael cipher and is based on substitution and permutation principles. The Rijndael cipher was invented by Joan Daemen and Vincent Rijmen. Due to a various number of factors that will be discussed later, AES is more secure and is quicker than DES.

4.3 A Flawed Symmetric Key Algorithm

Dr. Ayushi presented a symmetric key encryption algorithm in a research paper [1] titled “A Symmetric Key Cryptographic Algorithm”, which is simple and easy to implement. It uses 6 different steps for the encryption process.

1. The first step is to generate the ASCII value of the letter needing to be encrypted.
2. Step 2 is to generate the corresponding binary value of the letter. The binary should be 8 digits so for decimal 32 it would be 00100000.
3. Step 3 is to reverse the binary value.
4. Step 4 is to take the 4-digit divisor that is greater than or equal to 1000 as the key for the algorithm.
5. Step 5 is to divide the reversed binary with the divisor/key.
6. Step 6 is to store the remainder in the first 3 digits and quotient in the next 5 digits. If the remainder is less than 3 digits or the quotient is less than 5 then 0s will need to be added to reach 3 or 5 and this will form the ciphertext.

The decryption process is different than then encryption process and it only has 4 steps.

1. Step 1 is mto multiply the last 5 digits of the ciphertext by the key that was used to encrypt it.
2. Step 2 is to add the first 3 digits of the ciphertext with the results from step 1.
3. In step 3 if the result of step 2 is not an 8-bit number, 0’s will have to be added to make it one.

4. Step 4 is the reversal of the 8-bit value to obtain the plaintext.

This is an easy algorithm to implement, however, we have found this algorithm may not work for all possible keys described in the paper. For example, if you attempt to encrypt the letter “F”, which has the ASCII value of 70, with the key of 1101 you will end up with a remainder of 01010. This is a problem due to the remainder needing to be 3 digits for the ciphertext. The algorithm has a solution for if the remainder is less than 3 digits but not if it is more than 3.

CHAPTER 5: ASYMMETRIC KEY ENCRYPTION

The first type of encryption that was represented was conventional encryption, which is also referred to as symmetric key encryption. In symmetric encryptions, the key is shared between sender and receiver for both encryption and decryption. While in asymmetric key encryption, which is also referred to as public key encryption, a pair of keys is used, which are labeled as a public key and a private key. The public key can be shared to others and the private key is meant to stay hidden with the owner to keep the encryptions secure. Symmetric key encryptions involve the same key being used for both encryption and decryption and have algorithms that are based on permutation and substitutions. Public key encryptions are based on mathematics. However, neither symmetric key encryption nor asymmetric encryption is superior to one another. There are many different factors that go into determining which encryption is best for a specific scenario. The greatest hurdle when using asymmetric encryptions is the high computational complexity when applying the algorithm repeatedly to large amounts of plaintext. Asymmetric algorithms involve a lot of material derived from number theory that might be hard to understand without an extensive mathematics background. Every concept is not necessary to know to have a basic understanding of the entire concept. Public key encryption was invented to solve the two main problems that occur when using symmetric

key encryption, which are key distribution and digital signature. These will be discussed in further detail in a later section.

Here, in order to facilitate our discussion, we first introduce six main building blocks that are frequently used in public key encryptions.

1. Plain Text -- Plain text is the material that is going to be encrypted so that it is no longer readable without a form of decryption.
2. Encryption Algorithm -- An encryption algorithm is a function or mathematical change that is used on the plaintext to convert it into ciphertext that is no longer readable to the naked eye without a method of decryption.
3. Public and Private keys -- Public and Private keys are a “pair” of keys that are involved in the encryptions and decryption of information. One key is used to encrypt the information while the other is used to decrypt the ciphertext.
4. Ciphertext -- Ciphertext is what plaintext turns into after being ran through an encryption algorithm. Ciphertext changes based on the key and algorithm used. The same plaintext message can be outputted as two different ciphertexts depending on the key used when encrypting.
5. Decryption algorithms -- Decryption algorithms are used to turn ciphertext into readable plaintext. There are certain steps that public key encryptions follow. Two users wanting to send and receive messages between each other will both need to have a pair of public and private keys to be used. They both will need to share their public

key with one another and keep their private key just as the name states, private. When a user wants to encrypt a message to another, they will use the other person's public key to encrypt the plaintext and send it on over. The recipient can decrypt the received message with their private key. This allows for an additional layer of security because no one will have their private key so only the intended recipient will be allowed to decrypt the message.

Now that the basis of public key encryption has been described as well as a few examples of those widely used public key encryption algorithms, an asymmetric encryption algorithm will be introduced.

5.1 RSA

One of the first and most famous public key encryption algorithms is the Rivest-Shamir-Adleman algorithm which was published in 1978 and is commonly abbreviated as RSA. Since its release, it has been the most used general-purpose approach to asymmetric encryption. The basics of the algorithm are that for some n the plaintext and ciphertext are between 0 and $n-1$. n is usually 1024 bits. RSA encrypts plaintext into blocks and each block must have a binary value less than number n . This means that the size must be less than or equal to $\log_2(n) + 1$ and the block size must be i bits, where $2^i < n \leq 2^{i+1}$. For encryption and decryption, M is used as the plaintext block and C is used as the ciphertext block. These letters are used in the following equations for encryption and decryption.

$$C = M^e \bmod n$$
$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

Both the recipient and sender should know the value of n , and only the receiver knows the value of d and only the sender knows the value of e . This system makes RSA a public key encryption scheme that satisfies the requirements. The security of an encryption algorithm is very important to make sure certain information cannot be viewed by unintended recipients. One of the most common types of attacks is called a brute force attack. Which is when an attacker attempts to try every possible key until the correct one is selected. This is possible for simple encryption algorithms but the more complicated the algorithm the harder it is to brute force. The best way to prevent brute force attacks with the RSA algorithm is to increase the key size(d). As previously stated, this increase cannot be too large, or the computational power needed to encrypt and decrypt will be too much for practical use.

Another type of attack is called a timing attack. A timing attack is performed by monitoring how long it takes a certain block of ciphertext to be decrypted. Attackers can sometimes gain the private key by using timing attacks. There are few methods to prevent this attack from being successful. Some methods are used to increase the security of an algorithm. One of those examples is called constant exponentiation time. Which helps prevent the private key from being discovered by making it harder for specific values in the encryption to be discovered. This basically just means to make sure all the exponentiations take the same amount of time before the results are returned. This can increase security without decreasing performance drastically. Random delay is another method to help prevent an

algorithm from being analyzed. A random delay added at random intervals could throw off an attacker that is monitoring the decryption time. If enough random delays are not added, attackers can counter react the random delays and still discover some information. Binding is another act of attempting to make any attacks on the algorithm less likely to be successful.

Binding is the act of multiplying the ciphertext by a random number before it goes through exponentiation. This can cause the attacker to be unable to monitor each encryption bit by bit and will therefore make it harder to break the code. RSA Data security already naturally implements binding in practice. Binding can affect the RSA scheme performance by 2% – 10%. A common type of attack on RSA is called a fault-based attack. Fault based attacks are attacks on processors that are currently producing RSA digital signatures. The attack reduces power to the processor by inducing bit by bit errors until it starts to have invalid signatures, and this gives the attacker the chance they need to extract the private key used for the encryption. Chosen Ciphertext attack. A chosen ciphertext attack is when an attacker has a selection of ciphertext and their decrypted plaintext. They can use the information gained between the two to attempt to obtain the target's private key. This can be counter reacted by adding padding to the encrypted plaintext. The additional padding to the plaintext will need to be random and more than a simple padding. A simple padding could result in a complex attack still being able to be successful. Despite some of these security risks, RSA is secure and is used in many different applications today.

5.2 Diffie–Hellman Key Exchange

One of the first public key encryption algorithms was called the Diffie-Hellman key exchange or DH for short [7]. The algorithm is used to distribute a secret key that is shared among two users. This algorithm is useful because two users can agree on the shared key by communicating on channels that are not secure. The encryption algorithm steps start with a prime number ‘ q ’ and an integer ‘ a ’ that is a primitive root of ‘ q ’. The first user will choose a random integer that is $X_A < q$ and inputs it into the expression $Y_A = a^{X_A} \bmod q$. The second user will select a random integer as well that is $X_B < q$ and they will input it into the expression $Y_B = a^{X_B} \bmod q$. Each user will keep their X value private and can share their Y value on an insecure channel to each other. The first user will compute their key as $k = (Y_B)^{X_A} \bmod q$ and the second user will compute their key as $k = (Y_A)^{X_B} \bmod q$. Once both users have done the final computation, they will have a shared secret value.

5.3 ElGamal Encryption

One asymmetric key encryption algorithm based on the Diffie-Hellman key exchange is the ElGamal Encryption system [7], which is a cryptosystem that is made up of three components, which are key generator, encryption algorithm, and decryption algorithm. The cryptosystem is used between two users, similarly to the DH algorithm. The first user will take a multiplicative cyclic group G of order q with a generator of ‘ g ’. The first user will then choose a random number (x) from the set $\{0, \dots, q - 1\}$ and then they will compute $h = g^x$. The first user will use h and the description of G, q and g has their public key and will

keep 'x' as their private key and will not share it. This will complete the first step of key generation now that a public key and private key have been generated. The next step of encryption starts with the second user. The second user will need to choose a random y within the set $\{0, \dots, q-1\}$ and then they will calculate $c_1 = g^y$. The second user will then calculate their shared key with $s = h^y$. A new 's' value is computed for every message, so the s is also known as an ephemeral key. The second user will then convert their message(m) into an element m' of G and then will calculate $c_2 = m' \cdot s$. The second user will then send the ciphertext $(c_1 \cdot c_2) = (g^y, m' \cdot h^y) = (g^y, m' \cdot (g^x)^y)$ to the first user. This completes the step of encryption, and the last step is decryption. The first user will calculate $s = c_1^x$ and then they will compute $m' = c_2 \cdot s^{-1}$. Once they have computed m' they can convert it into m which is the plaintext message.

5.4 Elliptic-curve cryptography

Elliptic-curve cryptography (ECC) is a cryptosystem that is based on elliptic curves over finite fields [7]. ECC implements encryption, digital signatures, and key exchange. ECC is sometimes considered to be the successor to RSA. This claim is due to ECC using smaller keys and digital signatures than RSA. ECC provides the same level of security as RSA but has operations such as key generation, key agreement, and digital signatures, which are quicker than in RSA. Keys in ECC are usually 256-bit integers. Since ECC is based on elliptic curves, the key can vary depending on the curve it is being based on.

5.5 Two of the best-known uses of public key cryptography

5.5.1 Digital Signature

Digital Signature is an important aspect that is not present in conventional encryption. In asymmetric encryption, once a user has encrypted a message with their private key and sent it to another user, the message cannot be altered without the use of the original user's private key. This is a form of a "signature" that ensures that the message sent was indeed from the intended sender, which provides user authentication and acts as an additional layer of security to prevent an attacker from obtaining access to hidden information.

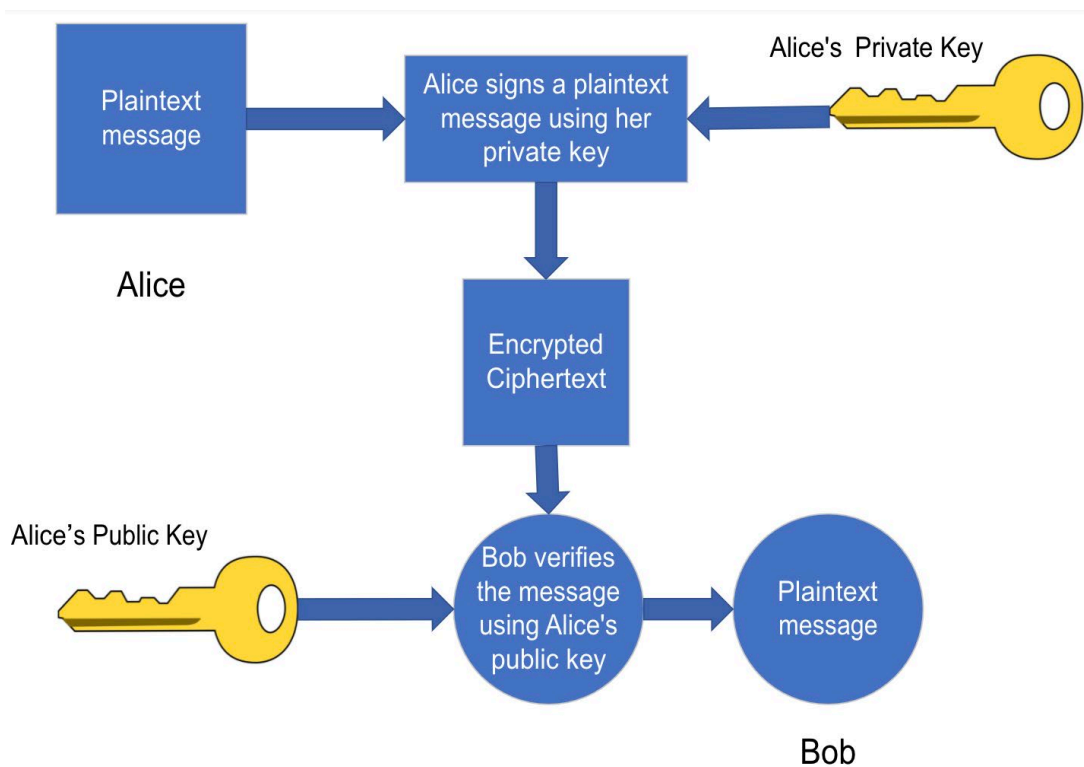


Figure 3. Representation of digital signature

In figure 3, digital signature is presented in a simple exchange between two hypothetical users, Alice and Bob. Alice and Bob are two common names used in examples in cryptography. In this example, Alice signs and encrypts a plaintext message that she is wanting to send to Bob with her private key. Bob receives the encrypted ciphertext and uses Alice's public key to determine if the message comes from Alice. If the public key can decrypt the ciphertext, Bob can trust that he is communicating with the authentic sender and does not have to worry about it being a man-in-the-middle attack. Otherwise, Bob can detect this message was not encrypted by Alice's private key or someone else has forged a message and pretend to be someone they are not.

5.5.2 Public Key Certification

In public key encryptions, there is a tool that can be used to determine if a person is the real owner of a public key. Public key certification is a document that contains information that can be used to determine the owner. The public key certification also includes the digital signature. If the digital signature is verified, a user can then trust the owner of the public key and they can perform secure communication between each other.

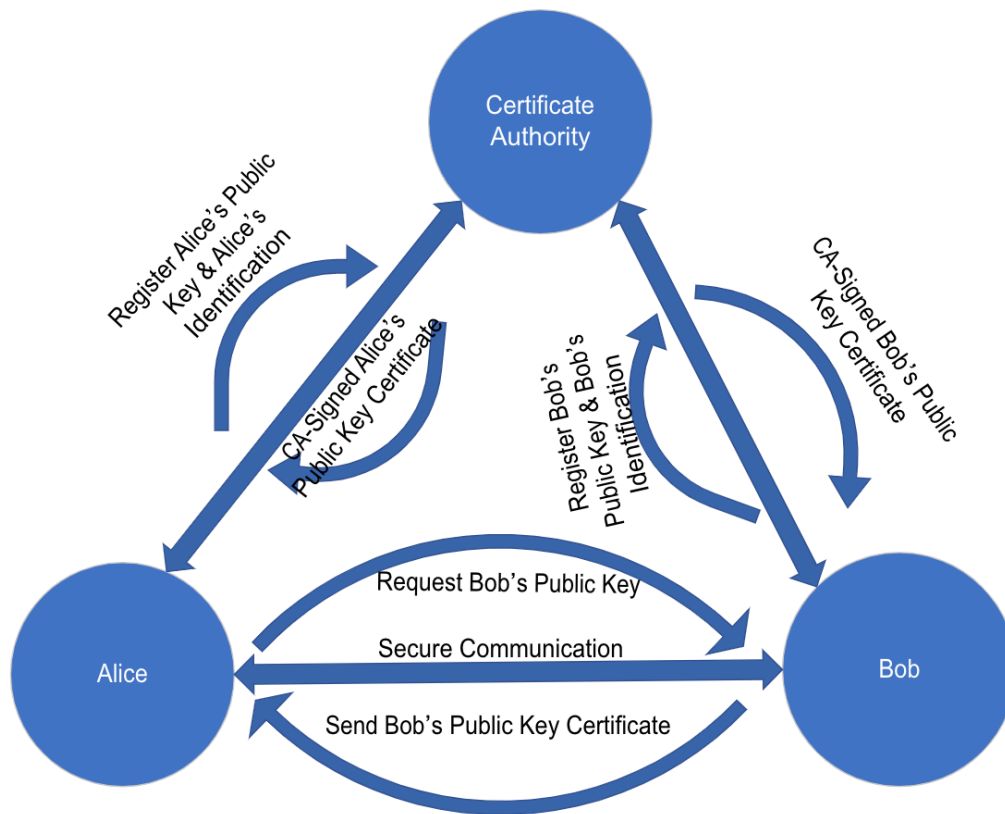


Figure 4. Public Key Certification

Figure 4 shows two users using public key certification to prove that each other is who they claim they are. Both users first need to register their information with a trusted third party, which is called certificate authority (CA). The certificate authority verifies both of their identification information and sends back a CA-signed certificate containing the corresponding requester's public key. Once a user needs to send information to the other, he or she first asks for the public key certificate and simply trusts they are who they say

they are. After the sender receives the receiver's public key certificate, the sender decrypts the certificate and obtains receiver's public key using CA's public key, which will be used for all encrypted messages sent to receiver from the sender in a secure manner. In this scenario it is assumed that Alice and Bob know CA's public key to decrypt the signed certificate.

CHAPTER 6: PERFORMANCE ANALYSIS AND COMPARISON

The environment used for the performance analysis is Python 3, the programming language that each algorithm was implemented in. For the implementation of the RSA algorithm, a python library called Crypto was used to aid the implementation. For timing each implementation, the python library time was imported, and a time variable was created at the beginning of the implementation. Another was taken immediately after the algorithm was implemented and the difference between the two times resulted in the total time the algorithm took to fully run.

Table 1. Performance Comparison among Five Encryption Algorithms

	DES	AES	RSA	Diffie-Hellman	Elgamal Encryption
SPEED	0.002178300	0.000048985	0.03258467	0.000017770	0.000889282
(seconds)	986997783	006287694	50056371	566046238	993739471
Key Size	56-bit	128-bit	1024-bit	G = 9 and P = 23	164-bit

Security Level	Low	Medium	High	Medium	High
-----------------------	-----	--------	------	--------	------

Table 2. RSA with Varying Key Sizes

RSA (Key Size)	1024	2048	4096
Speed (Seconds)	0.0242533683776855	0.0783960819244385	1.30468463897705

Table 1 shows the performance of various encryption algorithms described and the key size used when evaluating them. The key size for each is relatively small since it is for testing purposes and not for real world scenarios. Each algorithm is implemented and evaluated in Python. The speed row is precise up to 18 decimal digits to ensure an accurate test and comparison. Each encryption algorithm runs at a high level of speed, so the precision is needed to see the minute differences.

Table 2 shows the results of performing encryption and decryption using RSA encryption algorithm with varying key sizes. The results indicate a clear increase in time for each increment in key size. Every time when the key size is doubled, the computational time required is about doubled as well. The increase in security that comes from the increasing

key size is accompanied with an increase in computational time and computational power required to perform each encryption and decryption.

6.1 AES vs DES

The first two algorithms that will be compared are AES and DES. They are both very similar but also have a variety of differences. One of the first differences is that AES is based on a substitution-permutation network while DES is based on a Feistel cipher network. Another significant difference is the key size between the two, where DES uses 56-bit as a key size and AES can use either 128, 192, or 256-bit. The key size difference is one of the major reasons for AES being more secure than DES. As shown in the table above, AES outperforms in the speed category even with a larger key size, which means it is faster and more secure than DES. DES is no longer considered secure due to the small key size while AES is still considered secure. Due to the small size of DES's key, it can easily be brute forced and is not useable in most real-world environments for this reason.

6.2 AES vs RSA

The first two encryption algorithms to be compared are two symmetric key encryptions. Now one symmetric key encryption will be compared to a public key encryption algorithm. As seen in Table 1, RSA is much slower when compared to AES. This is due to a couple of factors with the main one being due to RSA having a much larger key size. RSA is mostly used to encrypt small amounts of information that need a very high security level. One downside of AES is that since it is a symmetric encryption algorithm, the encryptor

and decryptor must know the secret key. This can cause a security issue with managing the key and making sure it is not shared with anyone who should now have the access. Due to these qualities, AES is usually used for bulk encryption because of the high speed it offers. But when certain information is not as large and needs more security, RSA can be used. RSA and AES can also be combined to help take away the negatives of each algorithm. The way this would work would be by the key for AES being encrypted and decrypted via RSA so that the key management issue for AES is less of an issue and the speed problem with RSA is also managed due to it only being used for the key for AES. This is sometimes called a combined cryptosystem due to combining multiple encryption environments together in one.

6.3 Diffie-Hellman vs Elgamal Encryption

Instead of the Diffie-Hellman (DH) key exchange having a key size the size of G and P is usually used instead. In table 1, DH outperformed the Elgamal Encryption (EE) when it came to speed, but this was predictable. DH is used to generate a secret key that two users can then use in a symmetric encryption algorithm and it itself does not encrypt or decrypt any plaintext. Alternatively, the EE is used to do a similar task, but it also involves the encryption and decryption part, so it had more steps to do than the DH algorithm. Most of the time, the ciphertext that is produced by the EE is also a symmetric key that is used in algorithms such as AES. DH and EE are very similar due to the EE being based around the DH.

6.4 RSA vs DES

RSA and AES have been compared but not RSA and DES. DES has the speed advantage over RSA but when it comes to security RSA is the winner. As stated in the AES vs RSA comparison, sometimes hybrid cryptosystems can be formed to have the most benefits with the least number of consequences. RSA can be combined with DES but due to the low security of DES it would make the entire crypto system not very secure. Instead, RSA can form a hybrid cryptosystem with Triple DES (3DES). 3DES is just the DES encryption algorithm expanded 3 times to increase the security of the RSA-DES crypto system to a usable level. RSA would be used to send and receive the security key used in the 3DES algorithm and then the 3DES algorithm would be used to encrypt all data so that speed is not compromised. The security level and speed would still be worse than if AES was used instead [5].

6.5 Symmetric key vs Asymmetric key

When comparing asymmetric key encryptions with symmetric key encryptions the same conversations come up. Usually, asymmetric encryptions require a much higher computing power to encrypt information. While symmetric key encryptions are usually also very quick with the cost of a high security level. Most modern symmetric key encryptions are secure enough for practical use, but as technology progresses encryptions algorithms are replaced by newer ones and then the newer ones will eventually be replaced as well. This is like how DES was replaced by 3DES and then 3DES was replaced with AES. Asymmetric key

encryptions seem to be replaced less due to their high level of security and the difficulty to break them. Therefore, recently the two have started to be combined to perform the fastest most secure encryptions [7].

CHAPTER 7: CONCLUSION

After analyzing a variety of different algorithms and types of encryptions, there is a lot of information to conclude. Cryptography is a complex topic with a large mathematical background that can be difficult to understand for general audiences. The topics explored in this thesis go from the very basics of mathematical foundations and encryption algorithms all the way to the implementations of different encryption algorithms. The main two types of encryptions compared were symmetric key encryptions and asymmetric key encryptions.

Asymmetric and symmetric encryption algorithms are both very useful but have different performance results. Sometimes, one is better than the other in certain situations. Symmetric encryption algorithms are used more than asymmetric encryption algorithms due to the speed and computing power required for encryption. In symmetric key encryption algorithms, the speed is rapid, and the encryption does not require a lot of computing power. Symmetric key encryptions are usually used when a large amount of information needs to be encrypted or when security is not the biggest concern. In asymmetric key encryption algorithms, the speed is slower and more computing power is required due to the complexity of the encryption algorithms. Asymmetric encryptions have a higher security level, but the downside is the encryption speed. If security is the priority, asymmetric encryptions with large key lengths are the best to use but if speed needs to be

prioritized then symmetric key encryptions are superior. When a considerable amount of information needs to be encrypted but the security and speed are both a high priority, then both encryption methods can be combined. The key for the symmetric key encryptions can be generated with an asymmetric key encryption algorithm to provide additional security while using the faster encryption method for the bulk of the encryption.

As computer technology increases and computing power grows every year, the encryption methods that are researched and used has become more and more sophisticated and powerful. Some potential future research in cryptography may contain many different variations of both encryption methods discussed, and eventually a new encryption method altogether may be released. Future potential research topics for me include further analyzing of the most up to date encryption algorithms and implementing them to receive live data. After reading this thesis, readers should have the background knowledge to be able to perform their own encryption algorithm and to be able to decide which encryption algorithm is best for each specific scenario.

BIBLIOGRAPHY

- [1] Ayushi, "A Symmetric Key Cryptographic Algorithm," *International Journal of Computer Applications (0975 - 8887)*, vol. Volume 1 – No. 15, 2010.
- [2] A. Ochani, "DNA Image Encryption Using Modified Symmetric Key (MSK)," IEEE.
- [3] B. Kaliski, "A Survey of Encryption Standards," RSA Laboratories , 1993.
- [4] D. R. M. S.Suguna, "A STUDY ON SYMMETRIC AND ASYMMETRIC KEY ENCRYPTION ALGORITHMS," *International Research Journal of Engineering and Technology (IRJET)*, Volume 3, no. 4, 2016.
- [5] G. Singh, "A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security," *International Journal of Computer Applications (0975 – 8887)*, vol. Volume 67– No.19, 2013.
- [6] Mazhar Islam, "A New Symmetric Key Encryption Algorithm using Images as Secret Keys," Department of Telecommunication Engineering, University of Engineering and Technology Taxila, Pakistan. 2015.
- [7] M. Abdalla, M. Bellare and P. Rogawayz, "DHIES: An encryption scheme based on the Diffie-Hellman Problem," 2001.

- [8] M. A. Matin, M. M. Hossain, M. F. Islam, M. N. Islam and M. M. Hossain, "Performance evaluation of symmetric encryption algorithm in MANET and WLAN," 2009 International Conference for Technical Postgraduates (TECHPOS), Kuala Lumpur, 2009, pp. 1-4, doi: 10.1109/TECHPOS.2009.5412055.
- [9] Mandal, Bidisha, et al. "A Comparative and Analytical Study on Symmetric Key Cryptography." 2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE), 2014, doi:10.1109/icecce.2014.7086646.
- [10] S. Ahmad, "A Comparison between Symmetric and Asymmetric Key Encryption Algorithm based Decryption Mixnets," Graduate School of Engineering, University of Fukui. 2015.
- [11] S. Chandra, S. Paira, G. Sanyal and S. S. Alam, "A comparative survey of symmetric and asymmetric key cryptography," in International Conference on Electronics, Communication and Computational Engineering. 2014.
- [12] S. I. Bani Baker and A. H. Al-Hamami, "Novel Algorithm in Symmetric Encryption (NASE): Based on Feistel Cipher," 2017 International Conference on New Trends in Computing Sciences (ICTCS), Amman, 2017, pp. 191-196, doi:10.1109/ICTCS.2017.54.
- [13] Svetlin Nakov, "Practical Cryptography for Developers" Software University. 2018.
- [14] T. P. Innokentievich and M. V. Vasilevich, "The Evaluation of the Cryptographic Strength of Asymmetric Encryption Algorithms". 2017.

- [15] Thakur, Jawahar, and Nagesh Kumar. "DES, AES and Blowfish: Symmetric key cryptography algorithms simulation-based performance analysis." *International journal of emerging technology and advanced engineering* 1.2 (2011): pp. 6-12.
- [16] X. Luo, "Encryption algorithms comparisons for wireless networked sensors," Zhejiang Univeristy , Hangzhou, Zhejiang Province, China. 2004.