SECURING SENSITIVE BANK DATA BY USING ENCRYPTED ALGORITHMS. (AES, DES & RSA)

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

BY

SARA LUKA

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

JUNE, 2024

Approval sheet of the Thesis

This is to certify that we have read this thesis entitled **"Securing sensitive bank data by using encrypted algorithms. (DES, AES, RSA)"** and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Arban Uka
Head of Department
Date: June, 26, 2024

Examining Committee Members:

Prof. Dr. Bekir Karlik	(Computer Engineering)	________________
Prof. Dr. Betim Çiço	(Computer Engineering)	________________
Dr. Florenc Skuka	(Computer Engineering)	________________

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

Name Surname: Sara Luka

Signature: _______________

ABSTRACT

SECURING SENSITIVE BANK DATA BY USING ENCRYPTED ALGORITHMS. (AES, DES & RSA)

Luka, Sara

M.Sc., Department of Computer Engineering

Supervisor: Prof. Dr. Betim Çiço

With the increasing of technology and computer network, one major challenge or concern is confidentiality security and privacy. A potential way of securing authentication, identification and integrity is Cryptography. There are many techniques or aspects to transfer data safely through internet such as protection of passwords or secure payments transactions. Also one of these important techniques is even the cryptography to protect sensitive data. There are some techniques used in Cryptography which are used from Banks to ensure high security and privacy of data. Banks are always concern to communicate secret data/transactions through channel due to the security fact but Cryptography can assist in providing privacy while communicating important records/data.

This paper presents a fair comparison among three algorithms of Cryptography. Each algorithm uses a key to maintain the sensitive data and use it for decryption. This comparison will consist of different factors that will be analyzed in details to determine the most effective algorithm to be used for transferring sensitive and private data among different channels.

Keywords: *Encryption, Cipher text, Plaintext, AES, Blowfish, RSA; Twofish, RC2, DES*

ABSTRAKT

SIGURIMI I TE DHENAVE SENSITIVE TE BANKES DUKE PERDOUR ALGORITMAT E ENKRIPTIMIT.

(AES, DES & RSA)

Luka, Sara

Master Shkencor, Departamenti i Inxhinierisë Kompjuterike

Udhëheqësi: Prof. Dr. Betim Çiço

Me rritjen e teknologjisë dhe rrjetit kompjuterik, një sfidë apo shqetësim i madh është siguria e konfidencialitetit dhe privatësia. Një mënyrë e mundshme për të siguruar vërtetimin, identifikimin dhe integritetin është Kriptografia. Ka shumë teknika ose aspekte për të transferuar të dhënat në mënyrë të sigurt përmes internetit, si mbrojtja e fjalëkalimeve ose transaksionet e sigurta të pagesave. Gjithashtu një nga këto teknika të rëndësishme është edhe kriptografia për të mbrojtur të dhënat e ndjeshme. Ekzistojnë disa teknika të përdorura në Kriptografi të cilat përdoren nga bankat për të garantuar siguri të lartë dhe privatësi të të dhënave.

Ky punim paraqet një krahasim të drejtë midis tre algoritmeve të kriptografisë. Çdo algoritëm përdor një çelës për të ruajtur të dhënat e ndjeshme dhe për t'i përdorur ato për deshifrim. Ky krahasim do të përbëhet nga faktorë të ndryshëm që do të analizohen në detaje për të përcaktuar algoritmin më efektiv që do të përdoret për transferimin e të dhënave të ndjeshme dhe private midis kanaleve të ndryshme.

Fjalët kyçe: *Kriptimi, teksti shifror, teksti i thjeshtë, AES, Blowfish, RSA; Twofish, RC2, DES*

ACKNOWLEDGEMENTS

My time in graduate school was greatly enhanced by a multitude of individuals. I would want to thank my thesis supervisor and major professor, Betim Çiço, first. Working with him throughout the years has been intellectually stimulating and fulfilling. I would also like to express my gratitude to my sister, who influenced this study from the beginning of my dissertation work.

Many thanks to my friend, who patiently helped me with my word processing queries and issues. I also want to express my gratitude to my graduate school peers who supported me over the years I spent studying for tests and coursework.

My family deserves the final words of gratitude. I appreciate my brother, sister, and parents' patience

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In recent years, the growing information technology and field of communications has presented a big challenge to the world. Daily, computer networks and Google services grow and an extensive range of applications for example, online banking, e-commerce, e-government, and many others that assist in the public and private sectors. In order to receive data integrity and authentication of the data, these transactions and applications, which take place across wireless networks, must secure and thorough communications that protecting the process of network services.

In today's world, secure and protected communication is crucial as transactions and communications occur across public WI-FI. Robust processes are necessary for the management of these networks in order to authenticate data and ensure its integrity, privacy, and confidentiality. Strong cover is necessary to reduce the risk of threats and violations due to the volume of transactions and apps. Strong and efficient cyber security are a must to protect our digital infrastructure from hits and persistent threats. It is important to be aware of active threats that happened daily on internet services and to take some measurement in order to protect our sensitive data/information.

Cryptography is an important section of security and protection, providing essential methods and options to ensure confidentiality and integrity while data is being transmitted sent. Algorithms are part of cryptography because they make both the encryption and decryption procedures simple. Following up on the previous topic, asymmetric algorithms uses separate public and private keys for both decryption and encryption, in contrast to symmetric algorithms, which uses a single private key for both operations. If there is an option to choose between symmetric and asymmetric encryption there are many factors that should be considered as key management, and security requirements and computation efficiency. The current environment highlights the significance it is to use protection for establishing connections through networks and enable seamless electronic data transfer among different parties. In the meantime all of this interaction also increases the risk of attacks, illegal access to private data, and sensitive data sections. It is critical to take some approaches to cyber security that includes bringing together encryption algorithms with the purpose for reduce risk and any other intelligence threats. In substance, the lifetime goals of cryptography are to guarantee information security and visible information control and utilization. Targets of cryptography are to get to control data that goes through the internet services, monitoring these data and protecting them until they reach the receiver. Cryptography is common used to convert data into codes referring government intelligence all over the word and by doing that; it will secure the message transmission process that occurs online. Banks uses cryptography to keep the records and information very confident and secure. When a key is used to both encryption and decryption process, it means we are talking for symmetric algorithms. When we use two keys for both encryption and decryption process it is called asymmetric algorithm. Both processing are used to generate a cipher text from a plain text. Cryptography is the process of hiding sensitive information when receiver and sender communicated online. All the information that is shared via network can be private if it can go in the decryption process. Is always an open opportunity from attacks to threat the sensitive information online. Symmetric and asymmetric algorithms play a crucial role in this process to protect data. Symmetric algorithms are faster when it comes to performance.

What is Security & Privacy?

Table 1. 1 The difference between Privacy and Security

Privacy	**Security**
Using in an appropriate way user´s information. When the information is confidential and sensitive there should be a very careful care when using it.	Confidentiality, Integrity & Availability
Privacy means the decision of what information goes where. It is important to know where the sensitive information is going online and working on securing and protecting this information.	Confident and secure that information is respected. Security means protecting information from attacks and threats that goes online.
The issue here is safeguarding information from other parties	The purpose is to protect each party, the sender and the receiver.

CHAPTER 2

BACKGROUND

Cryptography includes two crucial types:

In symmetric cryptography process, both the sender and the receiver use the same key for the encrypted messages process. This process focuses on offering preferences, counting fast speed and visible security. By doing that, a disadvantage of symmetric key would be the need of safely transmitting the key to the other parties online.

In the meantime, asymmetric encryption, known as public-key, works by utilizing two different keys: the public key and the private key. The public key is broadly accessible from all parties, while the private key remains secret. Asymmetric algorithm differs from symmetric encryption, because it do not allow the uses of key from multiple parties. An advantage of public-key cryptography is that private keys are never transmitted, by improving security. Furthermore, this approach focuses in the creation of advanced utilization of both the private and public keys. In any instances, asymmetric encryption tends to be slower in comparison to symmetric encryption due to the computational overhead included in encryption forms.

Symmetric Encryption Algorithms

Due to the security highlights, symmetric encryption is separated as Triple DES (3DES), Data Encryption Standard (DES), and Progressed Encryption Standard (AES). Because of adaptability, the AES in specific is known as the de facto standard for symmetrical encryption. Numerous researchers have analyzed symmetric encryption algorithms' security and execution in detail, highlighting how successful they are in different circumstances, such as information encryption in cloud computing, arranging communications, and capacity frameworks. The key here should be provided in the communication process. Then the key will be divided into two blocks, block and stream chiper.

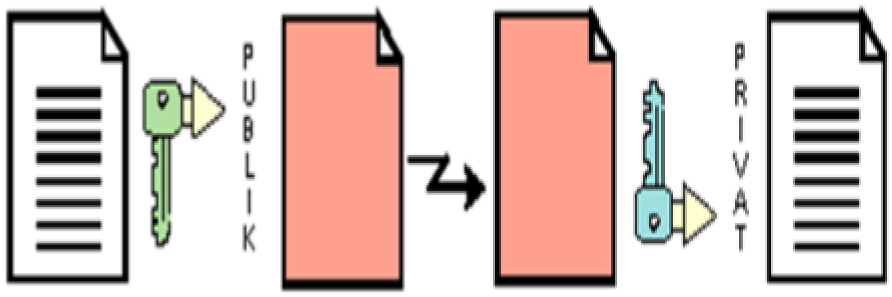


Figure 2. 1 Image of Public and Private Key

Asymmetric Encryption Algorithms

Public-key cryptography, every so often known to as asymmetric encryption, changed the range of cryptography by presenting a two key set concept, public and private keys. Elliptic Curve Cryptography (ECC) and the Rivest-Shamir-Adleman (RSA) are well-known asymmetric encryption. RSA is a very used asymmetric encryption, especially for securing computerized signs and key trade. In any case, ECC gives security with key sizes, which makes it perfect for resource-constrained situations like Web of Things. It is worth mentioning that a weakness of this algorithm is the key length which affects the process speed.

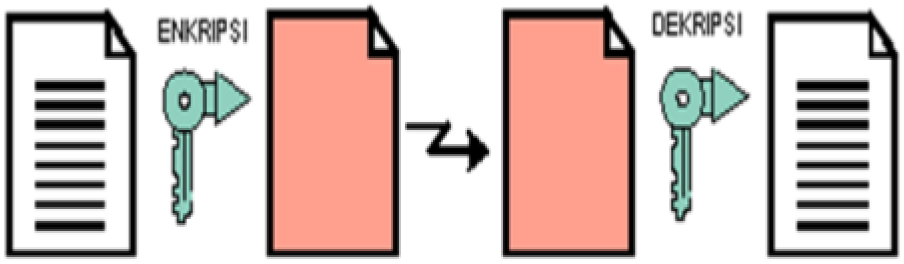


Figure 2. 2 Image of Public and Private Key

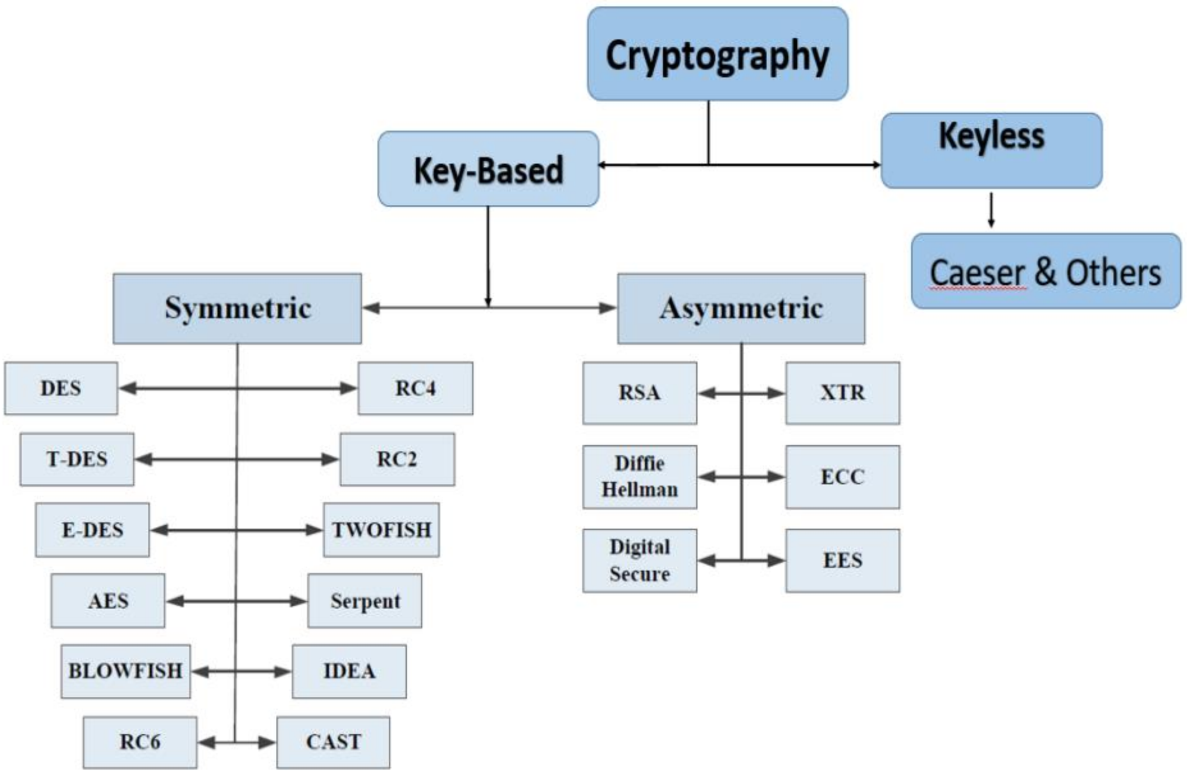


Figure 2. 3 Classification of Cryptography

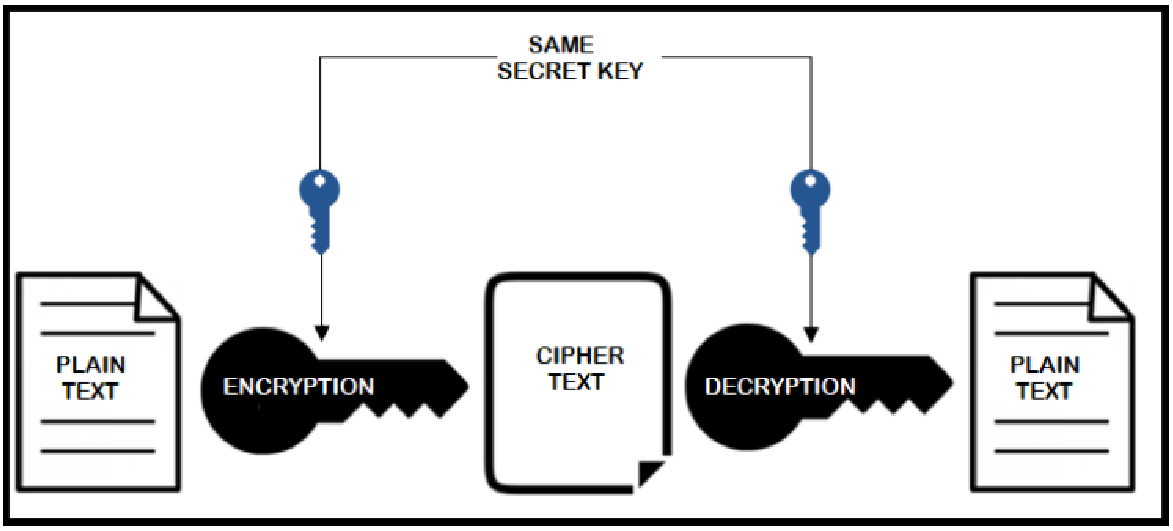


Figure 2. 4 Cryptography process

Encryption Algorithms Details

In this section we will have a further look of which are the most used algorithms today.

DES (Data Encryption Standard) this is one of the most used cryptographic algorithms used widely. It was created in 1977 and was first recommended by NIST (National Institute of Standard and Technology) DES may be a piece cipher which is outlined to encrypt and decode squares of data comprising of 64 bits by employing a 64-bit key. In spite of the fact that the input key for DES is 64 bits long, the real key utilized is as it were 56 bits in length. The slightest noteworthy bit in each byte is a equality bit. These equality bits are overlooked, so as it were the seven most noteworthy bits of each byte are utilized, coming about in a key length of 56 bits. The calculation goes through 16 Feistel rounds. The calculation changes 64-bit input in an arrangement of steps into a 64-bit yield. The same steps, with the same key are utilized for decryption. Numerous assaults recorded the shortcomings of DES made it an uncertain square cipher. In spite of the developing concerns almost its helplessness, DES is still broadly utilized by money related administrations and other businesses around the world for information security.

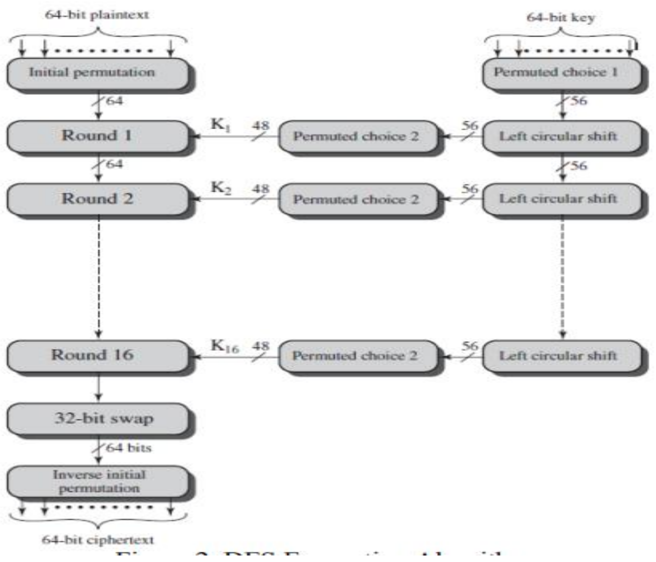


Figure 2. 5 Process of DES algorithm

3DES (Triple Data Encryption) was created by IBM in 1998-1999 to address the self-evident imperfections in DES without planning an entire unused cryptosystem. The 56- bit key utilized in DES isn't considered secure sufficient to scramble touchy information. 3DES basically expands the key estimate of DES by applying the calculation three times in progression with three distinctive keys. The combined key measure is in this way 168 bits. 3DES includes utilizing three 64-bit keys (K1, K2, K3) in Encrypt Decrypt- Encrypt (EDE) mold, that's , the plain content is scrambled with K1, then decoded with K2, and after that scrambled once more with K3: The encryption handle can be communicated as C = E(D (E (P, K1), K2), K3), additionally, decoding handle can be deciphered as P = D(E (D (C, K3), K2), K1)

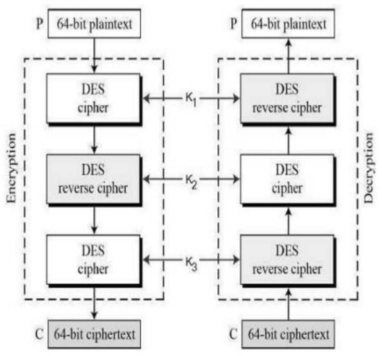


Figure 2. 6 Process of 3DES algorithm

Blowfish was outlined in 1993 by Bruce Schneier as a quick, free obvious elective to existing encryption calculations at that time. Blowfish gives a great encryption rate and no cryptanalysis of it has been recorded to date. The calculation comprises of two parts: a key-expansion portion and a data encryption portion. Blowfish employments an expansive number of sub keys which are developed amid the Key development stage. This stage changes over a variable-length key of at most 56 bytes (448 bits) into an cluster of sub-keys called P array comprises of eighteen 32-bit sub-keys totaling 4168 bytes. There are moreover four 32 bit S-boxes. The keys must be computed some time recently any information encryption or unscrambling. The creation of sub-keys encourage increments security, since a programmer would ought to break more than fair the initial key.

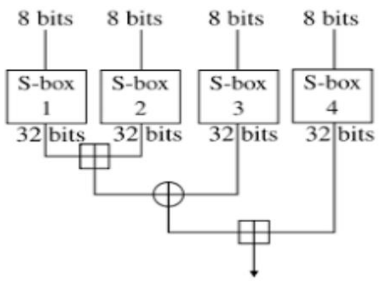
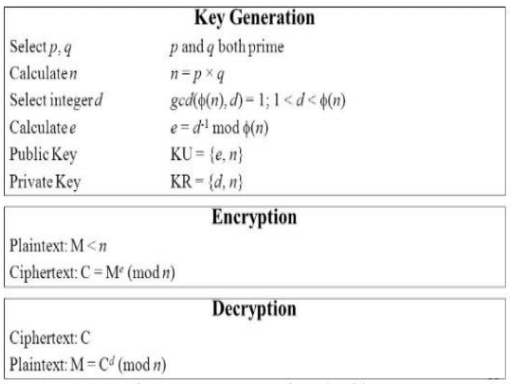


Figure 2. 7 Process of Blowfish algorithm

TWOFISH could be a symmetric block cipher with piece measure of 128 bits, and acknowledges a key of any length up to 256 bits. Two fish is quick on both 32-bit and 8-bit CPUs. It can be utilized in arrange applications where keys are changed as often as possible and where small or no RAM and ROM accessible. Two fish could be a Feistel organize. This implies that in each round, half of the content square is sent through an F work and after that XORed with the other half of the content square. In each circular of two fish, two 32-bit words serve as input into the work. Each word is broken up into four bytes. Those four bytes are sent through four diverse key-dependent S-boxes. The four yield bytes (the S-boxes have 8-bit input and yield) are combined employing a Most extreme Remove Divisible (MDS) framework and combined into a 32-bit word. Then the two 32-bit words are combined employing a Pseudo Hadamard Change (PHT), included to two circular sub keys, at that point XORed with the proper half of the text. There are moreover two 1-bit turns going on, one sometime recently and one after the XOR. Two fish too has something called "Pre whitening" and "Post whitening;" extra sub keys are XORed into the content square both sometime recently the primary circular and after the final circular.

RSA algorithm is first invented by Rivest and Shamir in 1979. This is one of the public keys encoding system used for digital signatures or encryption of blocks of database. This algorithm operates by using one key and one encryption block. Both plain text and cipher text are integrated between 1 and 0. One advantage of this algorithm is the high secure and safest algorithms. However this algorithm is not the best for speed performing. The three steps involved in this method are: the generation of keys, the encrypted process of plain text into cipher text and decrypted the data to the original.



Key Generation	
Select p, q	p and q both prime
Calculate n	$n = p \times q$
Select integer d	$gcd(\phi(n), d) = 1; 1 < d < \phi(n)$
Calculate e	$e = d^{-1} \bmod \phi(n)$
Public Key	$KU = \{e, n\}$
Private Key	$KR = \{d, n\}$

Encryption
Plaintext: $M < n$
Ciphertext: $C = M^e \pmod n$

Decryption
Ciphertext: C
Plaintext: $M = C^d \pmod n$

Figure 2. 8 Process of RSA algorithm

AES algorithm was first invented to replace DES & 3DES. This method owes a high level of secure and speed. This algorithm is crucial because it can be used in three sizes. AES is the only algorithm that consists of 3 sizes key. The data that is decrypted in 128 bits, take a 126 bits input and generate a 128 bits output by doing 10 rounds. Then the cipher text is executed. The data that is decrypted in 192 bits, takes 192 bits input and generate a 192 output by doing 12 rounds. And the last size type of AES is when the process takes 256 bits inputs and executes a cipher text of 256 bits by doing 14 rounds. This algorithm is called advance since it can processed files using different size types.

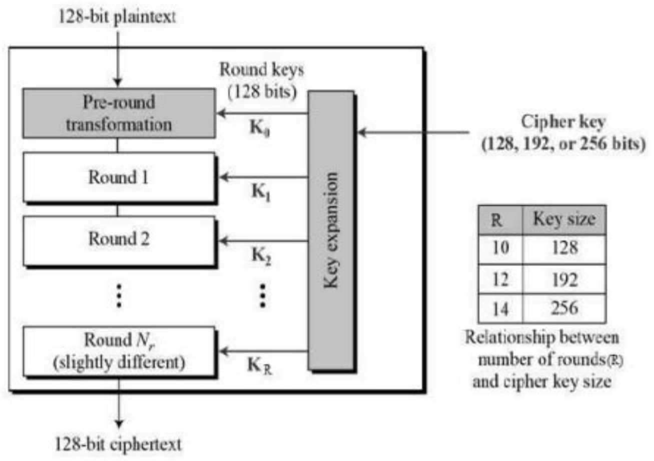


Figure 2. 9 Process of AES algorithm

Caesar cipher algorithm is a very used algorithm in cryptography. The Caesar process starts by shifting letters of the plaintext by a fixed number. The key for this algorithm is a single number which is mostly called shift. In the encryption process, his number will decide how many position each letter will shift in the alphabet. If key is 3, the letter A will be replaced by D. In the decryption process is the reverse process. The cipher text will be shifted backward by 3.

- Plaintext: "HELLO"
- Encryption: "HELLO" -> "KHOOR"
- Decryption: "KHOOR" -> "HELLO"

Basic Principles of Cryptography

Table 2. 1 Table showing the goals of the cryptography

Goal of Cryptography	**Description**
Data Integrity	Assurance that the data received is the same with the data sent to the receiver. It shows data consistency and correctness
Confidentiality	Ensure that the sensitive information is being sent to only authorize users.
Authentication	The verification and identity of the specific entity.
Non-Repudiation	Ensure that an entity cannot fail.
Availability	How ease if to access data or any other sensitive information
Access Controls	Permissions that are granted to specific users in order to access sensitive data or information.

CHAPTER 3

LITERATURE REVIEW

This literature review consists of analyzing around 50 papers where each paper is very well structured and their purposes are very well explained and implemented.
To start with a combination approach to encryption and decryption analyzed by the famous author. They focused on plaintext utilizing the 3DES algorithm and the keys utilizing the RSA calculation. Based on the execution, it limited RSA in asymmetric methods and DES, 3DES, Blowfish, in symmetric methods. However, RC6, and AES in symmetric methods, have been examined by author and his team. In the execution they secured a wide run of factors, such key trade, calculation adaptability, and security issues that are critical to consider when evaluating approaches for encryption. They found that Blowfish, RC6, and AES were the most secure and symmetric calculations, with high security levels. Also, they found that RSA was more proficient and effective than all symmetric algorithms used.

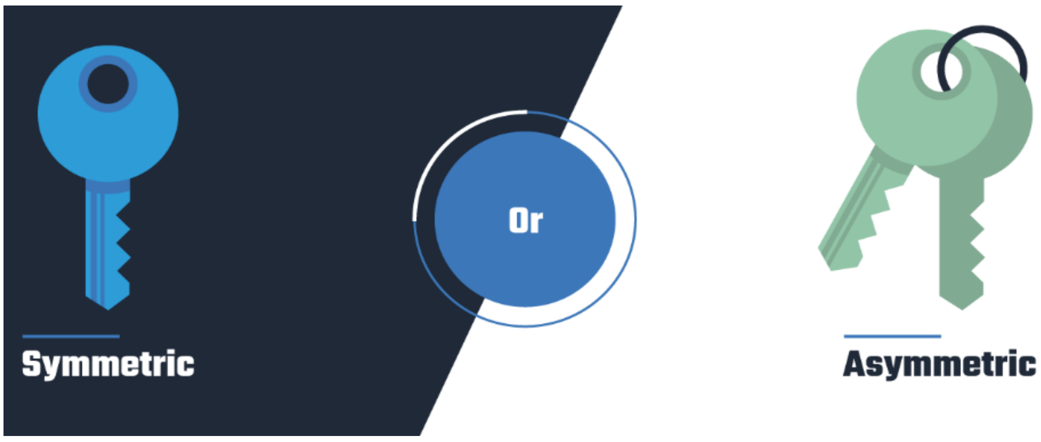


Figure 3. 1 Image of symmetric/asymmetric algorithm

According to the paper of the author [1] and her team, the most important algorithms to be compared are Blowfish, RC6, DES, AES & 3DES. They worked on comparing and securing a wide number of topics, counting details on key sizes, encryption/decryption speeds, battery control utilization, and contrasts in information square sizes. They concluded about a small variety between Base64 and hexadecimal base encodings. Based on the experiment, the results shown that aside from Blowfish, RC6 took less execution time. Time utilization was an continue issue for RC6, RC2, and Blowfish when comparing different information. They analyzed that 3DES performed not as well as DES algorithm. As it were the AES and RC6 calculations permitted for key measure alterations, which altogether modified battery and time utilization.

Using MATLAB software machine, author and co-authors [2] built popular symmetric encryption algorithms, such as DES, 3DES, AES, Blowfish, and RC4. Their comparison of implementations provided light on the avalanche effect caused by a change of one bit in plaintext. They discovered that RC4 showed the least amount of avalanche impact, while AES, DES, 3DES, and Blowfish showed the largest. The best option for producing a strong effect turned out to be AES algorithm where the mentioned effect is higher. Three cryptographic algorithms—DES, 3DES, and AES—were compared by the author [3]. The investigation considered different components, referring to block measure, security levels, key length, potential keys, and the sum of time required to check each conceivable key at a rate of 50 billion keys per moment. Based on the experiment, AES performed better than DES and 3DES. To maximize utilization; author [13] performed a comparative examination of AES and RC4. They tried different record sizes, from 100 Kb to 50 MB, utilizing measurements counting encryption/decryption time, throughput, CPU prepares time, and memory utilization. Their tests were shown in tablet with a 2.99 GHz CPU and 2GB Slam.

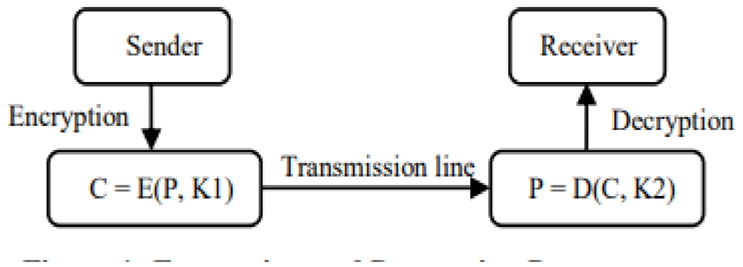
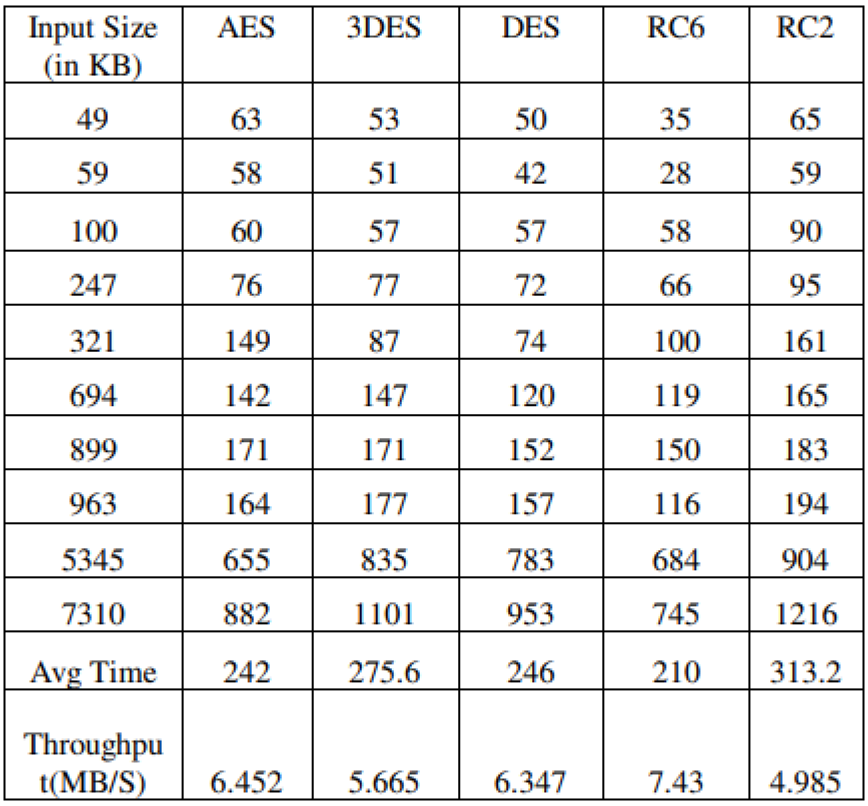


Figure 3. 2 Sender and Receiver data process

Author [48] and associates tested AES and Blowfish in order to conclude which of these two algorithms performed better for data protection. As part of their examination, they ran different encryption settings to handle shifting information amounts and measured how long it took the algorithm to prepare the information. Their findings resulted that Blowfish competed with less performance than AES since it required more computing control and time.



Input Size (in KB)	AES	3DES	DES	RC6	RC2
49	63	53	50	35	65
59	58	51	42	28	59
100	60	57	57	58	90
247	76	77	72	66	95
321	149	87	74	100	161
694	142	147	120	119	165
899	171	171	152	150	183
963	164	177	157	116	194
5345	655	835	783	684	904
7310	882	1101	953	745	1216
Avg Time	242	275.6	246	210	313.2
Throughpu t(MB/S)	6.452	5.665	6.347	7.43	4.985

Figure 3. 3 Comparative execution times (in ms) of decryption algorithm

An execution AES and Blowfish survey was carried out by author [4] and his team, who explored a few calculation parameters such key measure, square estimate, and security levels. They utilized a few exploratory conventions, such as base64 and hexadecimal encryption and decoding encoding methods, and shifted information bundle sizes, extending from 0.5MB to 20MB. Their experiment discovers that AES is better for high-security needs and is appropriate for picture encryption. In their paper of the DES and RSA algorithms, author [5] offered characteristics including encryption and decryption speed for text files of different sizes. Their results showed that DES performed better than RSA in terms of encryption and decryption times, with DES taking less time to execute. A comparison study of the security and performance of AES and RSA was carried out by the author [6] and his team, with an emphasis on encryption and decryption algorithms, mathematical elements, and security considerations. Their test results showed that AES is more secure and easier to implement than RSA, but that RSA requires more computing power because of its bigger key size. AES and RC4 calculations were compared by author [7], who set a solid accentuation on execution measures counting encryption/decryption time, throughput, CPU prepare time, and memory utilization. Based on their execution measurements comes about, they concluded that RC4 beats AES due to its speedy and energy-efficient encryption and unscrambling capabilities, which were bolstered by their exploratory discoveries

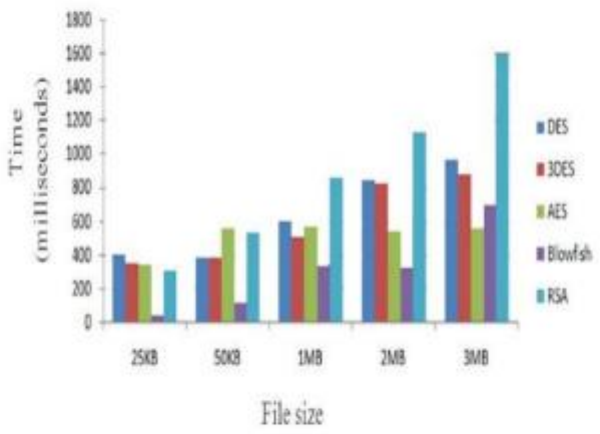


Figure 3. 4 Decryption time vs. file size for DES, 3DES, AES, Blowfish, RSA

A comparative examination of AES, DES, and RSA was carried out by [8]., with an accentuation on components like memory utilization and calculation time. Agreeing to their test comes about, RSA utilized more time and memory than AES and DES, though DES took the slightest sum of time to scramble information and AES utilized the slightest sum of memory.

Six common encryption calculations DES, AES, 3DES, Blowfish, RC6, and RC2—were evaluated by author and partners [9] taking under consideration factors counting information square sizes, information sorts, battery life, key sizes, and encryption/decryption speeds. Their come about appeared that RC6 was more viable than the other calculations, with the exemption of Blowfish. They too watched that when information sorts were changed from content to pictures, RC6, RC2, and Blowfish all appeared slower times. They take note that 3DES performed worse than the DES calculation which bigger key sizes had an impact on vitality and time utilization.

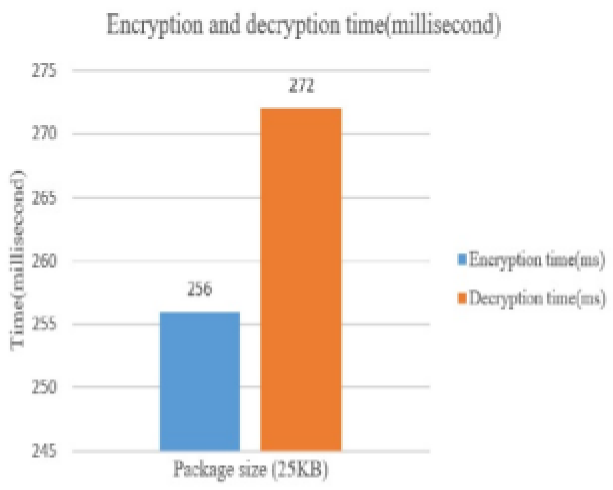


Figure 3. 5 Encryption and decryption time

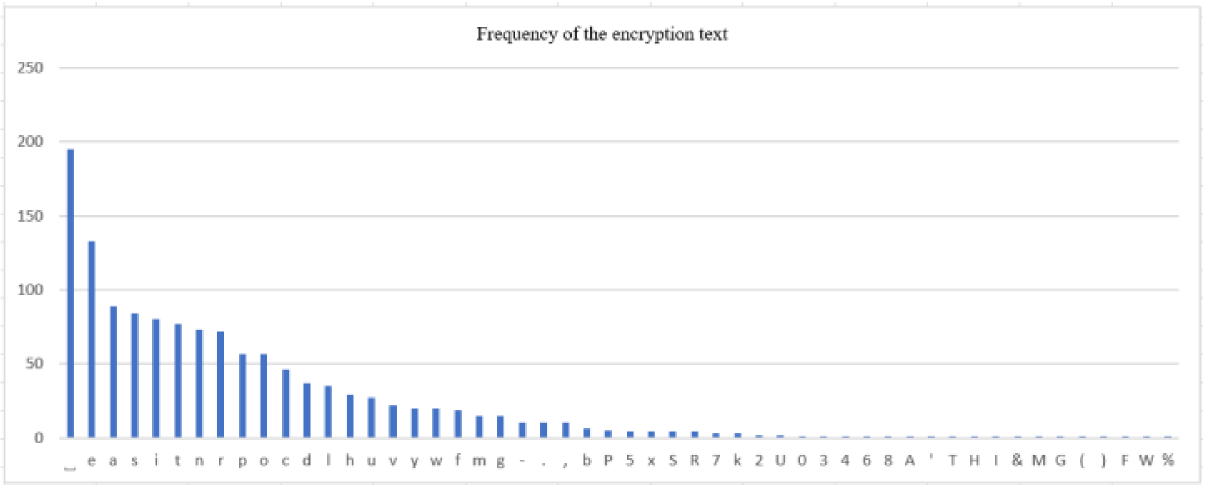


Figure 3. 6 Frequency of the encryption text

Asymmetric and symmetric cryptographic strategies were inspected by author [10] taking under consideration factors like square estimate and key length. AES outflanked DES and RSA in terms of encryption time, concurring to their appraisal utilizing an Intel Center i5 fourth-generation processor. AES and DES were compared for execution in terms of handling time, CPU utilization, and encryption throughput by author [12] and colleagues. AES was found to be speedier than DES in terms of execution time and throughput, whereas DES utilized less CPU. Their investigation was conducted on two diverse stages: an Apple MacBook with an Intel Center i5 processor and macOS, and a portable workstation Center i5 with a 2.5 GHz processor and Windows 7. Author [11] and colleagues compared the execution of DES, Triple DES, AES, and Blowfish calculations, utilizing distinctive record groups and sizes as input information. Their execution utilizing Java dialect and comparison on two equipment stages uncovered Blowfish as the speediest calculation in terms of execution, in spite of the fact that security issues were not considered in their tests.

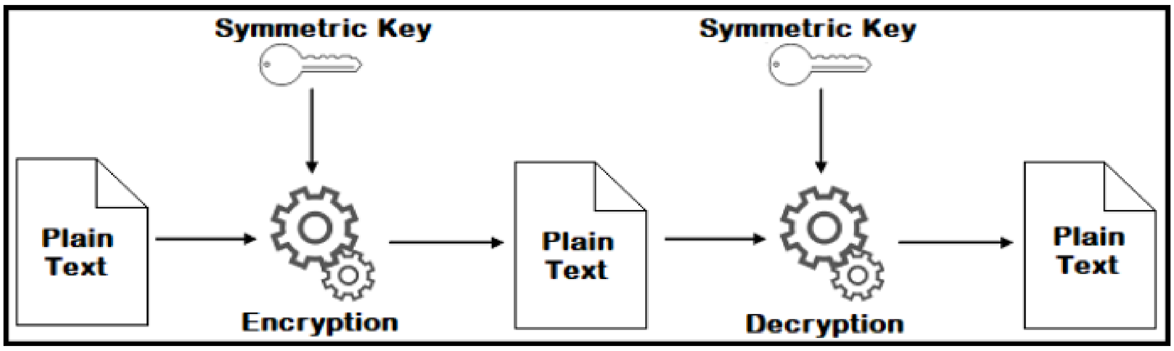


Figure 3. 7 Cryptography schema

In the following paper written by the author [15] a comparative study is done among different types of algorithms including symmetric and asymmetric. In this paper the author has used for the comparison the encryption and decryption time and the key generation size. The experiment is made in simple context to compare multiple algorithms. Another author [16] has analyzed different algorithms of cryptography by considering the following factors or parameters such as flexibility, security and key size. The purpose of this paper is to determine which algorithm will perform better and efficiently to protect data. The paper written by the author [17] has compare RSA, DES, DSA and AES, so algorithms from both symmetric and asymmetric algorithms. In this paper is analyzed the key length, the speed and the type of attacks that can affect these algorithms. The author [18] in the paper has discussed some algorithms methods and issues that are related with security. The techniques used in this paper are UMARAM and UR5 and are used to compare the architecture, security and limitation that are really important in cryptography. When it comes to the author [19] the paper represent an analytic analysis between DES and AES algorithm in order to generate and conclude which is the most effective algorithm to be used in data protection. The paper of the famous writer [20] has discussed for the most secure algorithm to be used in wired and wireless networks. Both symmetric and asymmetric algorithms have been analyzed in details. The following author [23] has analyzes in his paper both types of algorithms considering various setting of data. The aim of the author was to conclude the most efficient algorithm that is used in data protection. The following paper analyzes by the writer [24] represents a comparison among new encrypted algorithm in cloud computing. The paper goal was to identify and consider the key size, the performance of the encrypted process. The other paper written by the author [25] presents the mathematics process of RSA representing asymmetric algorithm and AES representing symmetric algorithm. The author [27] present in his paper a details comparison among Two fish and DES algorithm to identify which of the algorithms performed better. In the paper of the publisher [28] are discussed several cryptography methods that will conclude the cost of each algorithm in order to be used in data security and protection. The author and his colleges [30] chose ABE, ECC and IDEA to compare since they are not the most used algorithm in cryptography. This paper consists on analyzing two types of cryptanalysis, PTFT. The first section of the paper is a study for specific attacks of PTFT in different domains. The second section of the paper is proposed a new algorithm for attacks. The existing algorithm requires knowing a lot of information for keys in order to generate a successful testing or experiment. The paper of the publisher [31] has analyzed three algorithms of symmetric methods in order to generate which of them is more significant and accurate to be used is data protection. Another paper that is a good presentation of how the field of encryption written by [32] describes how fast is increasing every day. The comparison is based on each algorithm´s property. When talking for successful experiment, we talk for the paper of the famous author [33] which was focused on building a new method/algorithm to fight with existing processing so far. Another paper written, by the Chinese author shows how cryptography is applied in network. During the time the experimental paper of the author [35] was written, a potential survey of encryption algorithm was conducted. The survey includes the comparison of algorithms based on performance and amount of memory that each encryption method requires.

Another paper written by the author [36] has made an experiment of DNA image encryption. This method provides a secure layer included in symmetric algorithm by using MSK key. The following paper published by the author [38] is a comparison of algorithms but this time using an image as input in order to provide a new and original technique for encryption process. Another paper contributed in the encryption process by the author [39] where an comparison among asymmetric algorithms is done and concluded that RSA has a better performance. Another analytic comparison among DES and 3DES is made from the author [40] by representing a new novel in the history of cryptography.

DES (Data Encryption Standard)

DES is a 56 bit key which plays an important role in data security and data protection. The mentioned algorithm is very sensitive to attacks and most of the researches avoid using this algorithm when it comes to sensitive information that banks generates. Data in Des are encrypted in 64 bits blocks meaning that each input has the size of 64 bits and generate an output 64 bits as well. With a few small variations, the same key and algorithm are utilized for both encryption and decryption. The initial key is 64 bits but when the process is at the 8 bits section, the key is removed in order to generate a 56 bit key. This process will remove some bits resulting to a 56 bit key

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64

Figure 3. 8 Discording process of DES

The two core components of cryptography—substitution, commonly known as confusion, and transposition, often known as diffusion—are the foundation of DES. A round is one of the sixteen phases that make up DES. The stages of transposition and substitution are carried out in each round. Let´s discuss regarding DES steps

1. The 64-bit plain text block is given to a first Permutation (IP) function in the first phase.
2. On plain text, the first permutation is carried out. The first permutation (IP) then produces the Left Plain Text (LPT) and the Right Plain Text (RPT), which are the two parts of the permuted block.

3. Currently, the encryption procedure consists of 16 rounds for both LPT and RPT.
4. Finally, LPT and RPT are reconnected, and the combined block goes through to a Final Permutation (FP). This procedure provides 64-bit cipher text as its output.

Initial Permutation (IP)

As we mentioned, the initial permutation happen only after the first round. As seen in the example, it provides guidance on how the transposition in IP should be done. The text indicates, for instance, that the IP substitutes the 58th bit of the original plain text block for the first bit, the 50th bit for the second, and so on. For all other bit positions shown in the image, the same rule is applicable. After the PI is completed, the 64 bit text is divided into two blocks. Each block contains a 32 bits and each of the 16 bits consist of level steps as shown in the below image.

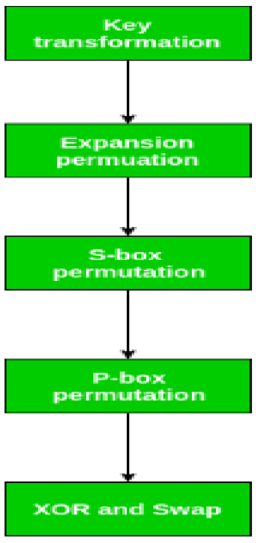


Figure 3. 9 DES stages

Let´s monitor in details each steps:

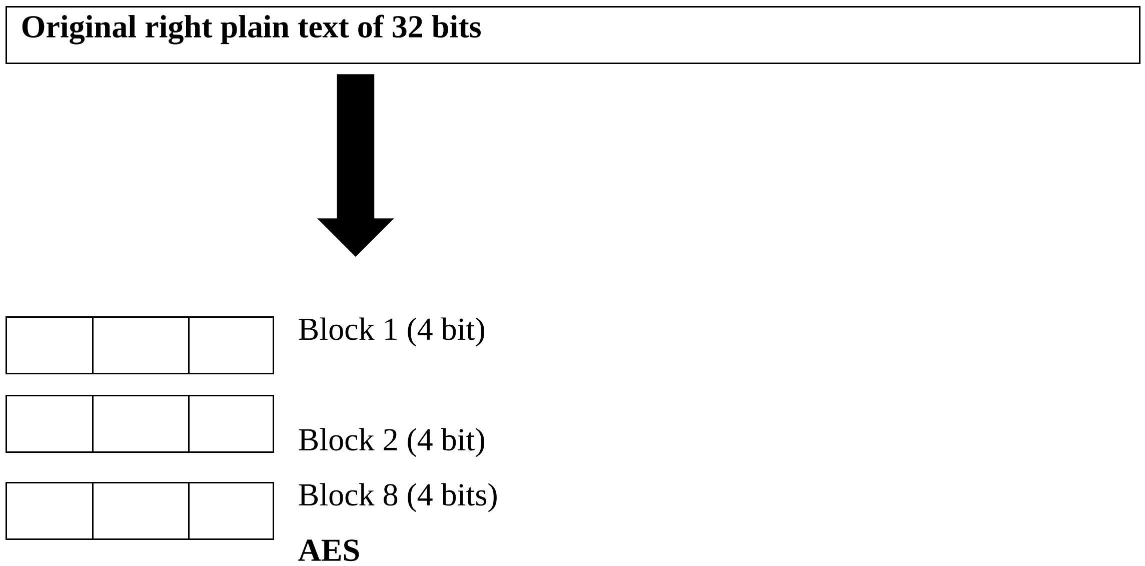
1. **Key transformation**
2. **Expansion Permutation**
3. **S- Box permutation**
4. **P – Box permutation**
5. **XOR and Swap**

Key Transformation

The initially generated 64-bit key is changed into a 56-bit key by removing each of the first 8 bits. Consequently, a 56-bit key is accessible for each. During each round, a distinct 48-bit Sub Key is generated from this 56-bit key using a procedure known as key transformation. The 56-bit key is split into two parts, each containing 28 bits, for this purpose. Depending on the round, these parts are circularly relocated left by one or two positions.

Expansion Permutation

Note that we had two 32-bit plain text parts named Left Plain Text PT) and Right Plain Text (RPT) following the first permutation. The expansion permutation causes the RPT to increase in size from 32 to 48 bits. The term "expansion permutation" refers to the permutation of bits as well. This occurs as a result of the 32-bit RPT being split up into 8 blocks, each of which has 4 bits. Next, each 4-bit block from the previous stage is expanded to a matching 6-bit block, meaning that two more bits are added for every 4-bit block.



AES

AES is the only algorithm that consists of 3 sizes key. The data that is decrypted in 128 bits, take a 126 bits input and generate a 128 bits output by doing 10 rounds. Then the cipher text is executed. The data that is decrypted in 192 bits, takes 192 bits input and generate a 192 output by doing 12 rounds. And the last size type of AES is when the process takes 256 bits inputs and executes a cipher text of 256 bits by doing 14 rounds. This algorithm is called advance since it can processed files using different size types.

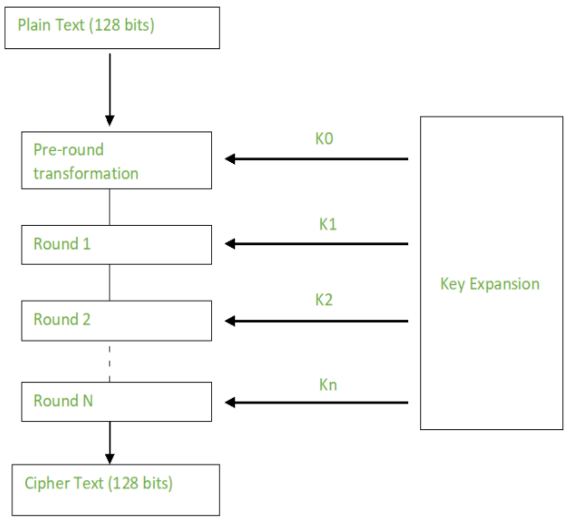


Figure 3. 10 AES Algorithm process

Each round has 4 steps.

Sub Bytes: In each step, bytes are substitute with another byte and it never substitute with itself. All the process is shown in a table called S-box and the output is a 16 byte matrix.

Shift Rows: In this step each row is shifted x times. The first row stays the same. The second row shifts once to the left. The third row is shifted twice to the left. The fourth row is shifted thrice in the left.

Mix Column: This step represents a matrix multiplication. Each column is multiple with a matrix.

Add Round Keys: In this step the output is XOR with the corresponding key.

Decryption process of the AES algorithm has 4 stages or steps which occur depending on the key size. The steps include Add round key, Inverse Mix Column, Shift Rows and Inverse Sub Bytes.

AES is used in many applications such as: Database encryption, secure communication, data storage, VPN, Secure storage and passwords. This paper will work on using decryption in protecting sensitive data included in the database.

RSA

RSA algorithm/method is an asymmetric process used in cryptography. RSA uses two keys, public and private keys. For example, a user sends a public key to the server and request data or sensitive information. The server will encrypt the data using the public key that user shared with and send the encrypted data. Then the user will receive the data and will decrypt it. SRA algorithm is secure and is very used for data transmission. This method is considered as safe because it uses two keys, the public one that can be shared online and private one that is used to decrypt the data. Also, both sender and receiver can exchange the key with each other and keep secure the private key. RSA is the best method used for digital signature meaning that a user can sign a document with the private key and the receiver can see the signature using the sender public key. RSA is very good method used in large application since it is quick and effective.

Some disadvantage of this algorithm is that it is slow in processing the data especially when processing a large amount of information or sensitive data. Also the key size of the RSA algorithm is very large which means that it needs more resources and spaces to be used. From different cases, it is concluded that RSA algorithm is very vulnerable from attracts online. RSA is not very common to be used in many applications.

3.1 State of Art

After reviewing and analyzing 45 papers we would separate 6 of them which are closely related with our topic. To start with the paper written by the author [41] on 2023 who has analyzed and compare two symmetric algorithms, Two Fish, Blowfish and AES. This comparison was based on key size, rounds, time and time consuming. From the comparison of these two algorithms, the author has concluded that AES is faster in encryption and decryption process. Also the experiment shows that 128 bit key has the shortest time to execute. Two Fish and Blowfish are lower in performance maybe because it is a standard algorithm that does not have the ability to execute faster and efficiently. In overall, AES is more suitable than Blowfish and Two fish.

Another paper that we would consider related with our topic would be the one written by the author [42] which is a comparison study of asymmetric algorithms, RSA and ECC. This paper was based on time consuming as well, the speed, block size rounds and decryption and encryption time. From the experiment, it is concluded that ECC is better when comparing with ECC in each factor. ECC had a better performance when it comes to speed. Also ECC is more suitable and appropriate when using in protecting and securing with lesser parameters.

The paper written by the author [43] and his colleges is a comparison among AES and 3DES. From the completed experiment, the author concluded that 3DES is more suitable for hardware system rather than software. It is very slow in comparing with AES and from the speed testing, the mentioned algorithm perform better. AES is an mathematical algorithm that its strength rely in the ability to have many key sizes, 128 bits, 192 bits and 256 bits.

Another similar paper we would consider is a comparison among DES, RSA and AES by the author [44] in 2020. This paper is only based on experiment tested by other authors and comparing which experiment is more efficient when using cryptography in sensitive bank data. From the detailed analysis made from the author, he has concluded that RSA is better when it comes to key size and is able to execute files, images and dataset in different sizes. Another parameter used in comparison is round. RSA needs 1 round to execute when AES and DES needs more rounds. When these methods are compared based on performance, DES and AES are more efficient and more suitable when comparing with RSA.

Another paper that is worth considering is written by the author [46] when he compare RSA and AES. After an experiment made by the author, he concluded that encoding and decoding process of AES is faster but it does not have a high level of security. From the other hand RSA is very unbroken toward attacks. RSA is not very fast in generating encrypted information but it is very protective. From the experiment system 1 has a poor performance and system 4 performs better. .

The following paper written by the author [45] on 2023 and published on 2024 compares also three algorithms, RSA and AES. The author has used Windows Azure SDK to form the application. When running the application with the help of cloud, a third party host, is measured. After reviewing the results, is concluded that AES is more secure than RSA and it also has a better performance. However this paper also suggests a hybrid method/model of AES and RSA which will bring another efficient method in the field of cryptography.

After the analysis of the papers written by other authors, the innovation that this paper represents is a comparison of DES, AES and RSA by considering more parameters that are needed to conclude which of the algorithms is better. The parameters are: encrypted and decrypted throughput, cost of each algorithm, time consuming, and security level, key sizes, memory, power consumption, battery consumption and attracts.

Also, what would be a good value for this comparison is the datasets that all of these algorithms will use. Both datasets will contain sensitive bank information and will contain a lot of records. By considering all the parameters and factors, we will be more clear which algorithm will perform better when it comes to protect and secure sensitive information that each bank contains. Also, another good aspect that is worth mentioning is how practical each algorithm is when we want to decrypt a file with large size.

Research GAP

While reviewing and analyzing 50 papers in the Literature review section, we concluded that there is a gap in the comparison of the algorithms. None of the mentioned papers have not compared the algorithms based on all the parameters. Furthermore, none of the analyzed papers have not tested algorithms using different datasets.

CHAPTER 4

HYPOTHESIS

H1: After analysis all the papers in literature review and state of art, our hypothesis for this master thesis is that AES will demonstrate higher level of security and performance compared to RSA and DES. While testing all the algorithms using different bank transaction dataset, we will prove and verify that AES is best for protecting data and sensitive bank information.

Research Questions

- Security:

  R0: How impacts have the key sizes of AES and RSA in their overall performance when testing method using bank dataset?
- Performance:

  R1: What are the complexities of both decryption and encryption process of three algorithms?
- Applicability
  - R2: Which of the three algorithms has a higher level of speed?
- Future Terms

  R3: Which of the current cryptography algorithms can be used for long terms secure?

The aim of this paper is to identify and verify which of the mentioned algorithm is most secure by considering the above research questions. The conclusion of this paper will contain results that shows that AES is more secure, has a high performance, is more practical to use and can be used in long terms.

CHAPTER 5

METHODOLOGY

The methodology used in this paper is **Quantitative Analysis**. It means that results will be generated by an **Experiment/Test** done in computer that will execute numerical information/data. This type of methodology will be completed by collecting data (**Datasets**) and testing it in the DES, AES and RSA algorithms in order to generate measurement outputs for the following parameters: encrypted and decrypted throughput, cost of each algorithm, time consuming, and security level, key sizes, memory, power consumption, battery consumption and attracts. By using this type of methodology, we will be able to prove, verify and answers the research questions. By answering the research questions made in Chapter 2, we will be able to confirm our hypothesis.

By using an experimental analysis approach, we focus more to present a systematic evidence and measurement of results. We aim to provide a comprehensive results and comparison through this experiment and data analysis in order to show the performance of each algorithm. Additionally the methodology used here will assist in validating our hypothesis and conclude effective conclusion and results in order to determine the most efficient algorithm to be used to securing sensitive data. Using the quantitative analysis and testing, will help us in comparing these algorithm in different scenarios while basing in different factors or parameters. With the generated results we can make our recommendations or suggested regarding the most effective algorithm to be used by banks in order to protect their data.

The quantitative analyses are used as a framework for evaluating cryptographic algorithms. This method is always able to provide a systematic approach to generate and execute reliable insight in the field of cryptography. Using this methodology, our goal would be to contribute in findings and inputs in the field of cryptography especially when we refer to secure sensitive information generated by banks.

5.1 Data Collection (Datasets)

We decided to source both dataset from the Kaggle website since this is a platform that deposits a large number of dataset. The datasets corresponds and are related with our research because they contain important bank records and are suitable for all the three algorithms that we will use for our experiment/test.

The first dataset contain 41189 rows and 13 columns including a comprehensive collection of transaction of bank users. The dataset has crusial columsn such as age, job, marital, education, contact, month, pdays, poutcome, em_variable, loan, month, duration and nr_employed. All of information will offer a very details description of users and how they interact with bank. The datasets focus more in personal data and presenting an important resource for analyzing and testing.

The second dataset which is smaller in number of records compared with the first dataset has 9234 rows and 12 columns. This specific dataset offer a varied collection of issues faces by each individual. The dataset shows directly the challenges that users encountered within the bank domain. All of these are perfectly displayed in each column that represents: product, issue, company, state, zip codes, Tags, submitted, timely resolution, consumer disputed and complaint ID.

The contrast in size among both dataset is a good opportunity to our experimental analysis. By doing this we aim to differentiate which algorithm should be used for large volume of data. That is the main reason why we choose two different size dataset to be compared in our paper. By using this strategy, we will better understand which algorithm is more suitable to be used for large amount of data and which is not very appropriate for large data collection. We will come to conclusion which algorithm will perform better. By performing we mean which of the three used algorithm in this dataset will be run faster than the other.

5.2 Libraries

Three algorithms code starts with the following library: **pip install pycryptodome**. This command installs **PyCroptodome** which is a library that contains methods and functions of cryptography. Furthermore this library will help us in implementing the required algorithm for this comparison. Once we run, download and install, the mentioned **PyCroptodome** library, we will be able to use all the cryptography Python functions or methods in our research. Once this library is installed, we will be able to import the necessary methods and functions in order to run the encryption, decryption or any other processes. The following line from **Crypto.Cipher** import DES, AES, RSA is important from the **PyCryptodome** library and can be used when we want to import the algorithm from the Crypto package. This module will help us in running various methods and function in algorithm.

Another module that is called from Crypto package can be found in the following line **from Crypto.Random import get_random_bytes**. Inside the mentioned package we can found the random module which imports the random bytes. By calling this line we will be able to generate random secure bytes that are essential for generating the key for all algorithms.

Another library used is **import pandas as pd**. This library can be used when we have already downloaded and installed the above library that we discussed previously. Panda is a source where we can use to call functions or methods related with data or datasets. Since in our research we will test the required algorithm using a very large dataset, this library is really important in order to collect, manipulate and analyze datasets with a large number of records. The pd is used to make the code more readable and easy and it alias with Panda library as a shortcut name. The following line is used in DES algorithm and we can see that pd library is used to read the uploaded dataset. **bank_transactions_df=pd.read_csv('/content/dataset/Dataset2MasterThesis.csv')**

Another library that we have used within the algorithm is **pip install psutil**. Pip is used to refer the package manager for Python that help in having the rights to access Python libraries. Psutil is library which is used to retrieve information related with CPU, memory, network. In our paper we will use this library to call the memory module which will help in our comparison among the three algorithms. In overall this library is used to access various system information or resources. This library is used in the following line used. The right part of the line is used to retrieve information for the virtual memory system and will generate information regarding how much is used the memory after running the algorithm (in bytes). The left part of the line is the variable that this function is assigned to. **memory_after_encryption = psutil.virtual_memory()**.

Another library used in three algorithms testing is **import time**. We need this library since we will compare the algorithms based on how long does an execution last. Time module includes the date, interval of time and it also measure the delays. A library used in this code is also **MatPlotLib.** This library can be called using the following line import **matplotlib.pyplot as plt. MatplotLib** is a library which contains plots and visualizations graphs and histograms. The histograms used in our code are imported from the **PyPlot** module within the **MatPlotLib.**

In the RSA algorithm code we have used three other modules. To start with the following line from **cryptography.hazmat.primitives import serialization, hashes**. A serialization is a module that can be used to serialize objects, in this case keys into byte sequence and desterilize them into original key. By using this module, we can convert the key to a format that can be stored and restored when it is needed. In our code we have use this module to concert RSA keys into PEM format.

The other module of the mentioned line is hashes. This module is used to generate functions by taking an input and producing a fix size string. Another module used in RSA algorithm is shown in the following line: **from cryptography.hazmat.primitives.asymmetric import padding, rsa.** The asymmetric module make it valid to call functions that involve two keys and separate them which is used to encryption and which one is used for decryption. In the asymmetric library we can import the padding module. Padding is like a policy for data inputs that ensure that information meets requirements. For example, OAEP and PKCS are used in RSA in order to add security features.

Another module that can be import from the initial library is shown in the following line: **from cryptography.hazmat.backends import default_backend**. This module is useful because it contains functions that ensure that each implementation is suitable for the cryptography processes and functions.

5.3 Environment

In order to execute and complete this experiment, we used the Google Colab environment for all the cryptography processes and operations. This environment is very suitable to run various libraries and packages Python based including all the cryptography libraries used for all the operations and processes. Using this environment will make the experiment easier as it does not need any setup of installation. Furthermore, Colab offers GPU acceleration which can help us in intense processes or methods. All the cryptography functions and processes are implemented using Python 3.8 and importing the cryptography library version 3.4.7.

The experiment conducted on a computer environment with the following parameters/specifications:

System: Linux

Node Name: 89e4fb814ca4

Release: 6.1.58+

Version: #1 SMP PREEMPT_DYNAMIC

Machine: 86x64

Processor: 86x64

5.4 Visualization

An important aspect of our paper is visualization our results. We conducted that visualizing our findings, will offer a better understanding of the comparison we have decided to test and run. As explain in the Library section, we have imported the Matplotlib module which is a popular Python library to create the visualization part of our testing. Now let´s explain the following lines that have assist us on visualizing our findings. The below code has been used in our experiment to show on histogram the encryption and decryption throughput, encryption and decryption time and encryption and decryption memory consumption.

```
itime_values = [function1, function2]
x_labels = ['Function 1', 'Function 2´]
plt.figure(figsize=(x, y))
plt.bar(x_labels, y_values, color=['xcolor', 'ycolor'])
plt.title(Function1 & Function 2')
plt.xlabel('Process')
plt.ylabel('Time (x)')
plt.grid()
plt.show()
```

The first line of the code shows a list that generate two functions which are stores in function 1 and function 2 examples. The x_labels shows the labels for the x-axis of the histogram that represent two functions called Function1 and Function2.

The next line of the code plt.figure(figsize=(x, y)), is used to put the size of the figure to x width and y height. By using this line it will result in having some specific dimensions when the histogram is shown.

The bar plot shown in the following line plt.bar(x_labels, y_values, color=['xcolor', 'ycolor']), create the histogram with the specified labels and values. Also both bars have a specified color in order to be separated from each other.

The next line of the code is useful since it shows the title of the histogram. It is used to show the purpose of the histogram and what is the histogram representing. We have set the title based on what we are generating such as encryption and decryption throughput, encryption and decryption time and encryption and decryption memory consumption.

If we are referring to the next line plt.xlabel('Process'), it means that we are adding a label to the x´-axis indicating the process we want to show. The nest line interpret and specify the unit of measurement in the label y. The last two lines assist in our purpose as it add grid lines to the histogram and make it easier to read the data and the plt.show displays the histogram.

This experiment visualization consists of two, Histogram and Scatter plot. These two types of visualization will help us interspersing better the results and making it readable from the others.

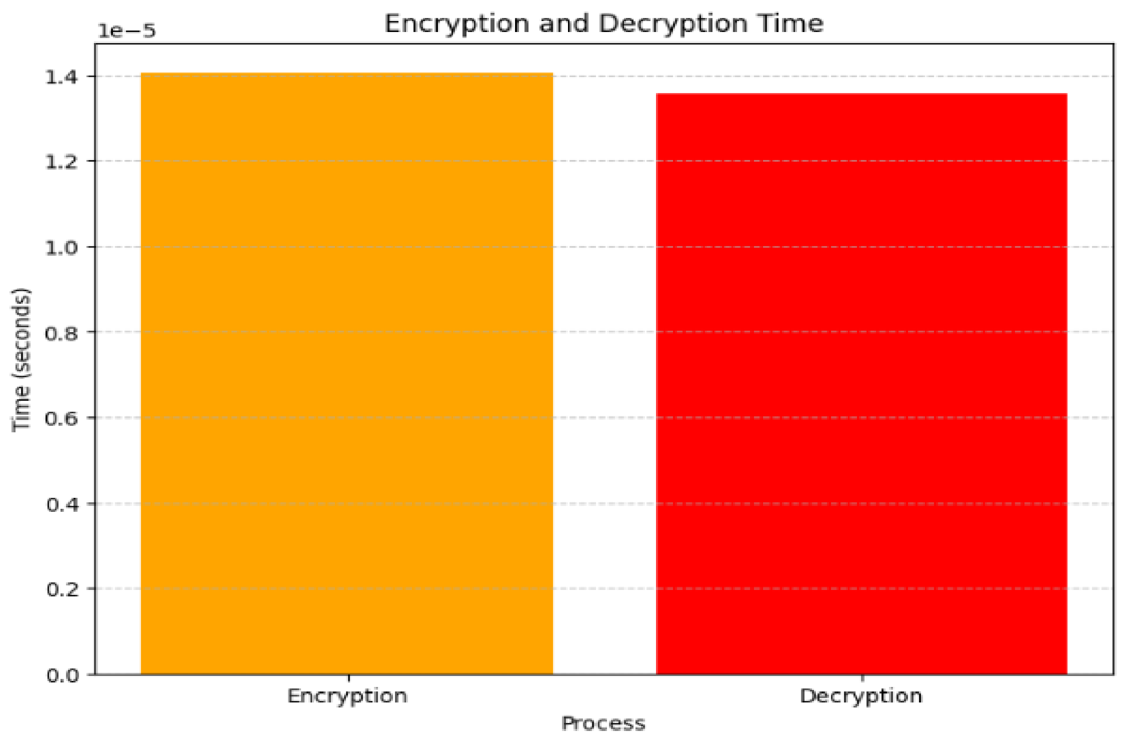
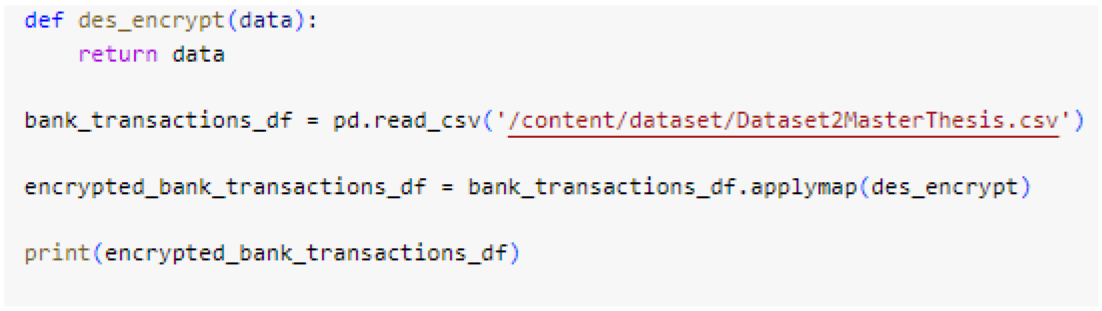


Figure 5. 1 DES Encryption and Decryption Time

CHAPTER 6

EXPERIMENT

This experiment is made by using functions/methods in order to help us in comparing DES, AES and RSA. In this section we will have a comprehensive look of each method and how are these methods representing with histograms.



```
def des_encrypt(data):
    return data

bank_transactions_df = pd.read_csv('/content/dataset/Dataset2MasterThesis.csv')

encrypted_bank_transactions_df = bank_transactions_df.applymap(des_encrypt)

print(encrypted_bank_transactions_df)
```

Code used to read and print the dataset

6.1 DES Encrypted and Decrypted Method

Below we will show an example of how the data has been encrypted using DES algorithm.

The below column shows the product data

Product

Bank account or service

Bank account or service

Bank account or service

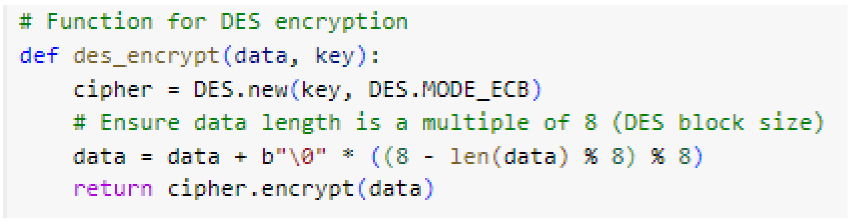
Bank account or service

The below column shows the product data that are already encrypted

Product

```
b'u\xa2\xe2\xa4S\x1f\xc1wY\n\xc9plO\xae<I\x937...
b'u\xa2\xe2\xa4S\x1f\xc1wY\n\xc9plO\xae<I\x937...
b'u\xa2\xe2\xa4S\x1f\xc1wY\n\xc9plO\xae<I\x937...
b'u\xa2\xe2\xa4S\x1f\xc1wY\n\xc9plO\xae<I\x937...
b'u\xa2\xe2\xa4S\x1f\xc1wY\n\xc9plO\xae<I\x937...
```

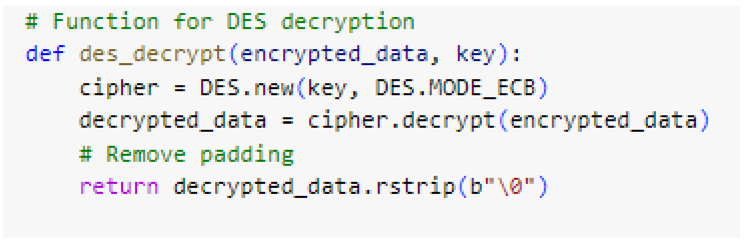
For this experiment, one of the most important functions/methods used are the encryption and decryption processes. We will have a further look at every functions or method used for this testing. The below code is captured form the DES algorithm and it perform the encryption process.



```python
# Function for DES encryption
def des_encrypt(data, key):
    cipher = DES.new(key, DES.MODE_ECB)
    # Ensure data length is a multiple of 8 (DES block size)
    data = data + b"\0" * ((8 - len(data) % 8) % 8)
    return cipher.encrypt(data)
```

The above code takes as parameters data and key. The data is the plaintext which will be encrypted and key is the secret one to be used for the encryption purposes. The second line of the above code initialize the cipher object with the mentioned key and ECB which is a block cipher mode where each plaintext block is encrypted independently. What would be important to add is also the length of the data validation. The third line of the code ensures that the length of data is also multiple with 8 bytes which is the size of each DES block. The code then will encrypt the data using DES cipher and it will return the cipher text.

The code below is an example of Des decrypted method that we have used for our experiment. This function takes the key and the encrypted data returned from the previous method.



```python
# Function for DES decryption
def des_decrypt(encrypted_data, key):
    cipher = DES.new(key, DES.MODE_ECB)
    decrypted_data = cipher.decrypt(encrypted_data)
    # Remove padding
    return decrypted_data.rstrip(b"\0")
```

As mentioned, the function takes as parameters the key and the plaintext. Then it do the same action by initializing the Des cipher object using the key and the ECB mode. The following line shows the method that can be used after importing the necessary library explains the Library section to decrypt the data. During this process, the function will remove the padding which was initially added to ensure that data is multiple with 8. Then the final step is to return the decrypted data to the receiver.

Encrypted the Entire Dataset Function

The below code will help is our testing in order to encrypted the entire dataset. As mentioned in the Data Collection section, for this algorithm we have used a dataset with approximately 9000 records.

```
encrypted_bank_transactions_df = bank_transactions_df.applymap(lambda x:
des_encrypt(str(x).encode(), key))
```

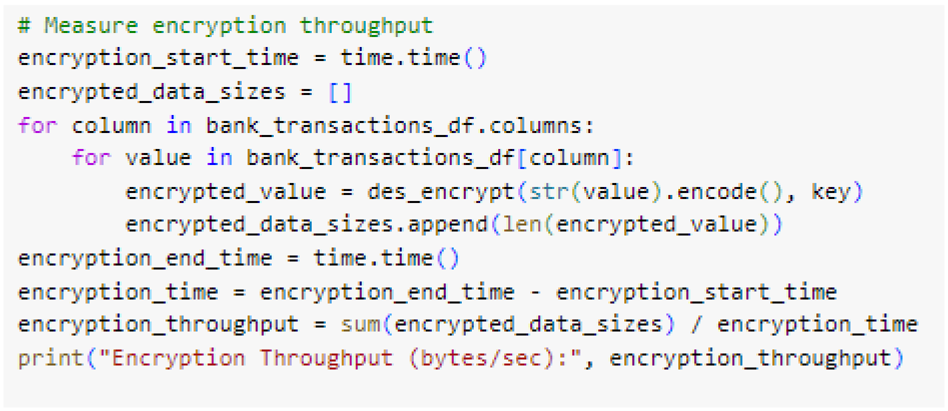
The code represents the bank_transactions_df as a data frame for each element of the DES encrypted function. Applymap is a module imported from Pandas that is applied to each data of the data frame. Also in the code we see the lambda function. This method has several functions such as:

1. It converts each element x of the data frame to a string. str(x)
2. It encode the string as bytes by using the following function .encode()
3. It encrypts each byte using the des function and the key
4. encrypted_bank_transactions_df would be the result of Data Freame were all the elements are encrypted using Des function and the secret key.

Overall, this line of code will encrypt each record in the bank_transaction_df data frame using the DES algorithm and the key, then it store all the encrypted data in a new data frame called encrypted_bank_transactions_df.

DES Encryption Throughput Method

An important factor that will help us in comparing these algorithms is how is the performance of each algorithm in the cryptography processes while testing them with full records datasets.



```
# Measure encryption throughput
encryption_start_time = time.time()
encrypted_data_sizes = []
for column in bank_transactions_df.columns:
    for value in bank_transactions_df[column]:
        encrypted_value = des_encrypt(str(value).encode(), key)
        encrypted_data_sizes.append(len(encrypted_value))
encryption_end_time = time.time()
encryption_time = encryption_end_time - encryption_start_time
encryption_throughput = sum(encrypted_data_sizes) / encryption_time
print("Encryption Throughput (bytes/sec):", encryption_throughput)
```

What is needed for this function is the time method that we can use it after importing the necessary libraries. The first line of the code helps in recording the current time when the encryption process starts. Then the For loop starts by iterating each column of the Data frame. Each value is converted to a string, then they are encoding to bytes and then encrypted using the DES encrypted function with the private key. The next line is used to specify the length of the encrypted value in the list. Then the time function will record the end time of the process then we calculate the total time taken for the whole process.

The following line **encryption_throughput = sum(encrypted_data_sizes) / encryption_time** calculate the throughput of the process by dividing the total size of data by the encrypted time. It shows the rate at which data is encrypted measured in bytes/sec. Then we use the print method to print the throughput of the encrypted process. To summarize, the function is used in order to have a better understanding and view on how fast the records in the data frame can be encrypted measured in bytes per seconds.

The below code has been used to calculate the throughput of the decrypted process. It shows the same process used in the encrypted process but this time we will take into consideration the decrypted process calculated in the first part of the code. This function generates the time method that we can use it after importing the necessary libraries. The first line of the code helps in recording the current time when the decryption process starts. Then the For loop starts by iterating each column of the Data frame. Each value is converted to a string, then they are encoding to bytes and then decrypted using the DES decrypted function with the private key. The next line is used to specify the length of the decrypted value in the list. Then the time function will record the end time of the process then we calculate the total time taken for the whole process.

The following line **decryption_throughput = sum(decrypted_data_sizes) / decryption_time** calculate the throughput of the process by dividing the total size of data by the encrypted time. It shows the rate at which data is encrypted measured in bytes/sec. Then we use the print method to print the throughput of the encrypted process. To summarize, the function is used in order to have a better understanding and view on how fast the records in the data frame can be encrypted measured in bytes per seconds.

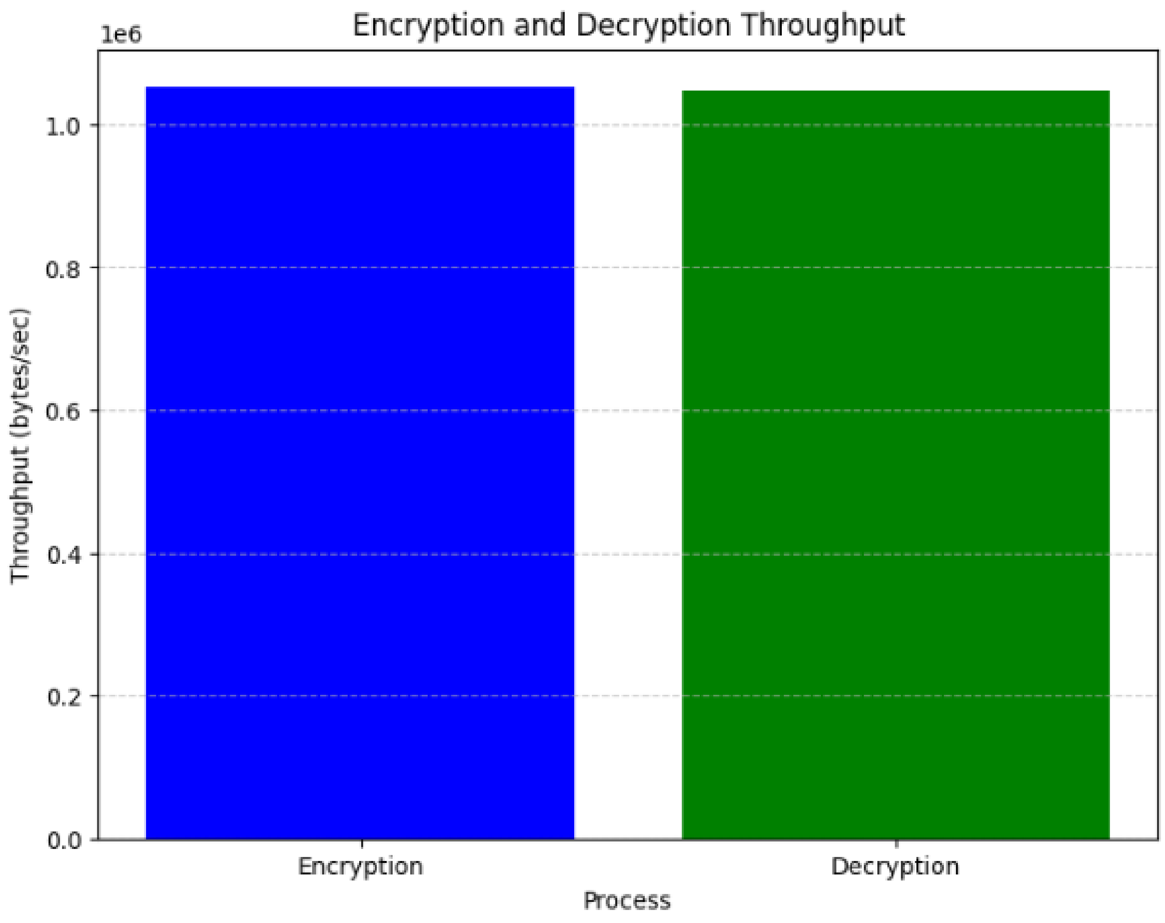


Figure 6. 1 DES Encryption and Decryption Time

Memory Usage after Encrypted Method

Another important parameter that will help us in comparing the algorithm is definitely the memory usage. The below code has been used in DES algorithm to show the rate of the memory usage when executing this algorithm.

```
memory_after_encryption = psutil.virtual_memory().used
encryption_memory_consumption = memory_after_encryption - initial_memory
print("Memory consumed during encryption (bytes):",
encryption_memory_consumption)
```

To start with the first line of the above code, it uses the psutil library that we have mentioned in the Library section and it is used to sow the amount of memory used. The next line of the use will calculate the memory usage with an easy calculation. We differentiate the memory after the encryption process with the initial memory. Then the code will execute the results shown in bytes. Now we will calculate the memory usage after the decrypted process. The below code has been used in DES algorithm to show the rate of the memory usage when executing this algorithm. The same calculations are done in this section as well but this differentiates because now we will take into consideration the decrypted process. Then the code will calculate the memory usage after the decrypted process in bytes.

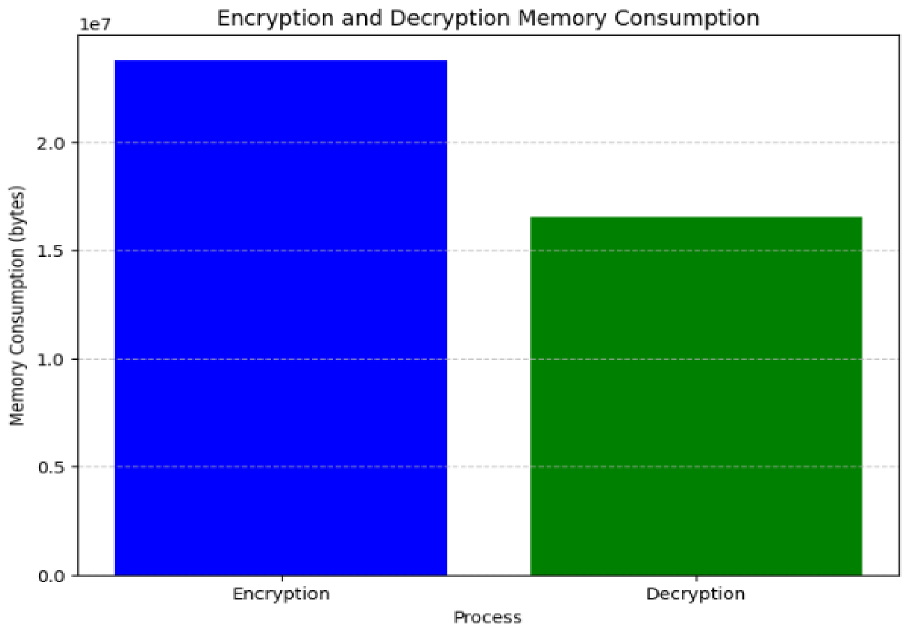


Figure 6. 2 DES Encryption and Decryption Memory Consumption

6.2 AES

Now let´s take a look at the methods and functions that we have used for testing the AES algorithm. For this experiment, one of the most important functions/methods used are the encryption and decryption processes. We will have a further look at every functions or method used for this testing. The below code is captured form the AES algorithm and it perform the encryption process.

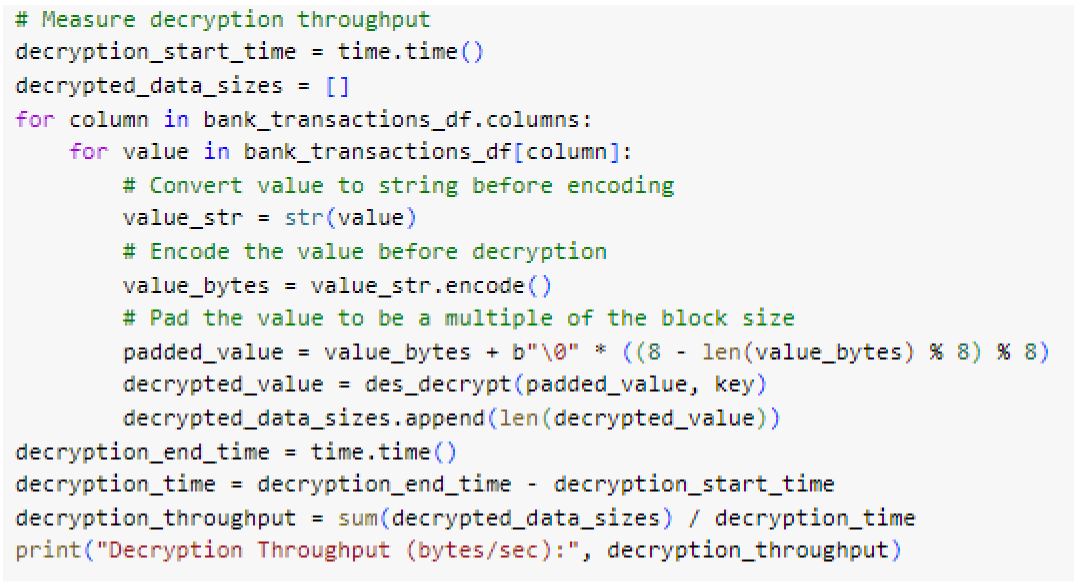
```
def aes_encrypt(data, keyy):
    cipher = AES.new(key, AES.MODE_ECB)
    data = data + b"\0" * ((16 - len(data) % 16) % 16)
    en_data = cipher.encrypt(data)
    return en_data
```

The above code takes as parameters data and key. The data is the plaintext which will be encrypted and key is the secret one to be used for the encryption purposes. The second line of the above code initialize the cipher object with the mentioned key and ECB which is a block cipher mode where each plaintext block is encrypted independently. What would be important to add is also the length of the data validation. The third line of the code ensures that the length of data is also multiple with 16 bytes which is the size of each AES block. The code then will encrypt the data using DES cipher and it will return the cipher text.

AES Encryption Throughput Method

In comparing these algorithms is how is the performance of each algorithm in the cryptography processes while testing them with full records datasets we will need the encrypted throughput. We need the time method that we can use it after importing the necessary libraries. The first line of the code helps in recording the current time when the encryption process starts. Then the For loop starts by iterating each column of the Data frame. Each value is converted to a string, then they are encoding to bytes and then encrypted using the DES encrypted function with the private key. The next line is used to specify the length of the encrypted value in the list. Then the time function will record the end time of the process then we calculate the total time taken for the whole process.

The following line **en_throughput = sum(en_data_sizes) / en_time** calculate the throughput of the process by dividing the total size of data by the encrypted time. It shows the rate at which data is encrypted measured in bytes/sec. Then we use the print method to print the throughput of the encrypted process. To summarize, the function is used in order to have a better understanding and view on how fast the records in the data frame can be encrypted measured in bytes per seconds. We have used the below code to calculate the throughput of the decrypted process. It shows the same process used in the encrypted process but this time we will take into consideration the decrypted process calculated in the first part of the code.



```
# Measure decryption throughput
decryption_start_time = time.time()
decrypted_data_sizes = []
for column in bank_transactions_df.columns:
    for value in bank_transactions_df[column]:
        # Convert value to string before encoding
        value_str = str(value)
        # Encode the value before decryption
        value_bytes = value_str.encode()
        # Pad the value to be a multiple of the block size
        padded_value = value_bytes + b"\0" * ((8 - len(value_bytes) % 8) % 8)
        decrypted_value = des_decrypt(padded_value, key)
        decrypted_data_sizes.append(len(decrypted_value))
decryption_end_time = time.time()
decryption_time = decryption_end_time - decryption_start_time
decryption_throughput = sum(decrypted_data_sizes) / decryption_time
print("Decryption Throughput (bytes/sec):", decryption_throughput)
```

This function generates the time method that we can use it after importing the necessary libraries. The first line of the code helps in recording the current time when the decryption process starts. Then the For loop starts by iterating each column of the Data frame. Each value is converted to a string, then they are encoding to bytes and then decrypted using the AES decrypted function with the private key. The next line is used to specify the length of the decrypted value in the list. Then the time function will record the end time of the process then we calculate the total time taken for the whole process.

The following line **de_throughput = sum(de_data_sizes) / de_time** calculate the throughput of the process by dividing the total size of data by the encrypted time. It shows the rate at which data is encrypted measured in bytes/sec. Then we use the print method to print the throughput of the encrypted process. To summarize, the function is used in order to have a better understanding and view on how fast the records in the data frame can be encrypted measured in bytes per seconds.

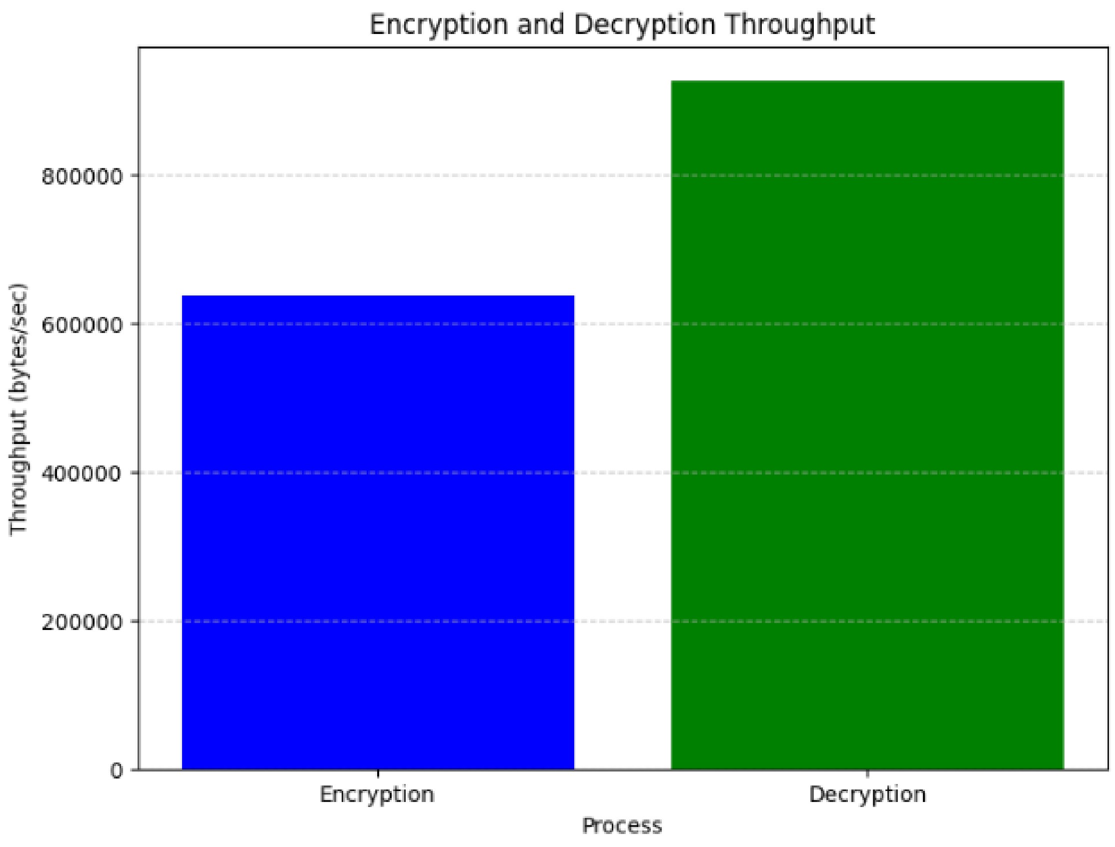


Figure 6. 3 AES Encryption and Decryption Throughput

Memory Usage after Encrypted Method

Another important parameter that will help us in comparing the algorithm is definitely the memory usage. The below code has been used in AES algorithm to show the rate of the memory usage when executing this algorithm.

```
i_memory = psutil.virtual_memory().used
for column in bank_transactions_df.columns:
    for value in bank_transactions_df[column]:
        aes_encrypt(str(value).encode(), key)
memory_after_en = psutil.virtual_memory().used
en_memory_consumption = memory_after_en - i_memory
print("Encryption Memory Consumption:", en_memory_consumption)
```

To start with the first line of the above code, it uses the psutil library that we have mentioned in the Library section and it is used to sow the amount of memory used. The next line of the use will calculate the memory usage with an easy calculation. We differentiate the memory after the encryption process with the initial memory. Then the code will execute the results shown in bytes.

Memory Usage after Decrypted Method

Now we will calculate the memory usage after the decrypted process. The below code has been used in DES algorithm to show the rate of the memory usage when executing this algorithm.

```
# Get memory usage after decryption
memory_after_decryption = psutil.virtual_memory().used
decryption_memory_consumption = max(memory_after_decryption -
memory_after_encryption, 0)
print("Memory consumed during decryption (bytes):",
decryption_memory_consumption)

# Plot histograms for memory consumption
memory_values = [encryption_memory_consumption,
decryption_memory_consumption]
memory_labels = ['Encryption', 'Decryption']
plt.figure(figsize=(12, 6))
```

The same calculations are done in this section as well but this differentiates because now we will take into consideration the decrypted process. Then the code will calculate the memory usage after the decrypted process in bytes.

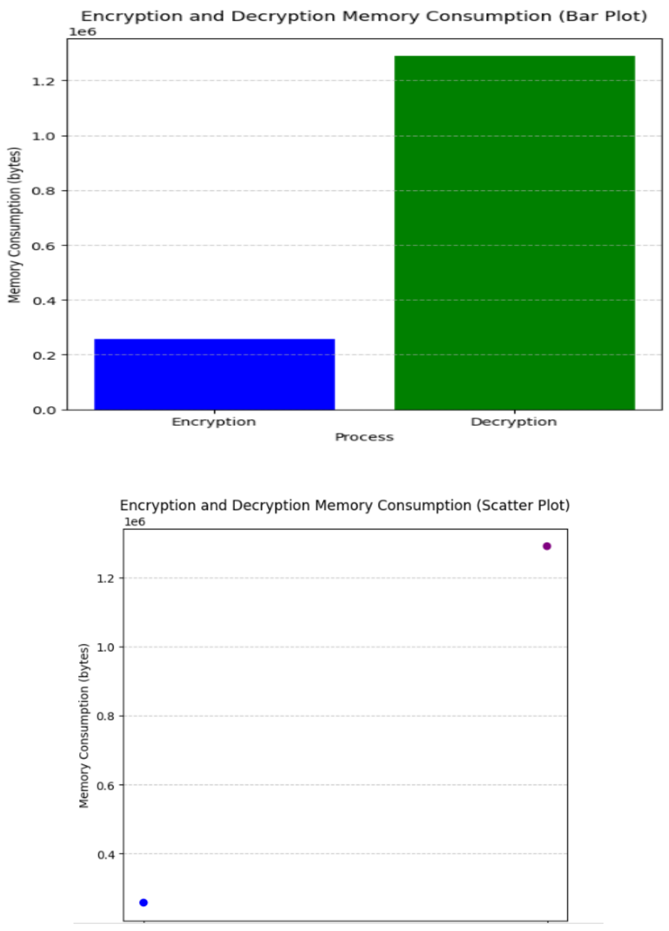


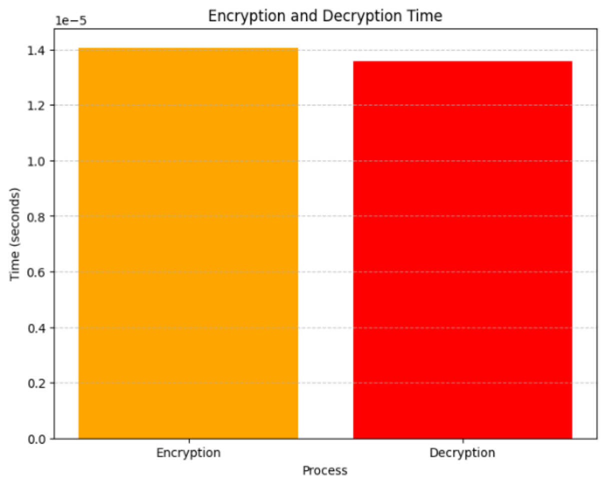
Figure 6. 4 AES Encryption and Decryption Memory consumption

AES Calculated Time Method

Another important factor that will help in our comparison is Time. The following code is an example used in our testing. The first line of the code records the starting time of the process, in this case the encrypted process. Then the for loop iterate each column and in each column the nested loop iterate the records. The following step of the process is encoding each string to bytes and then encrypts them. The last line calculates the time of the process by differentiating the end time from start time.

```
decryption_times = []
encrypted_bank_transactions_df = bank_transactions_df.applymap(lambda x:
des_encrypt(str(x).encode(), key))
for _, column in encrypted_bank_transactions_df.items():
```

```
for encrypted_data, _ in column:
    _, decryption_time = des_decrypt(encrypted_data, key)
    decryption_times.append(decryption_time)
```



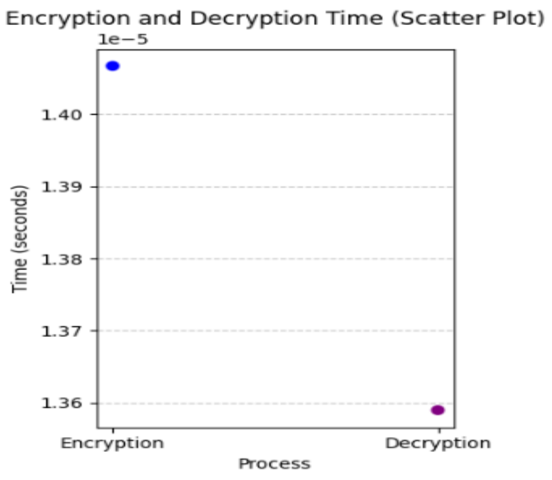


Figure 6. 5 AES Encryption and Decryption Time

6.3 RSA

This section will introduce to the functions and methods used to test this algorithm. To start with the code used to execute the whole dataset that consists of 4100 rows and 21 columns. This part of the code will perform the key. The public export set to 65537 is a parameter that specifies the public key in RSA process. It is often chosen as 65537. The key size is set to 2048 and it means that the key will be 2048 bits long. The back end parameter specifies the backend process using the key. When we run this code, the private key will be generated as a key object that will be used for the encryption and decryption processes. However generating an RSA key doesn´t mean that it will be used to interact with the dataset but there will be created a pair key that will be used for encrypting the dataset.

RSA private key code

```
   encoding=serialization.Encoding.PEM,
   format=serialization.PrivateFormat.PKCS8,
   encryption_algorithm=serialization.NoEncryption()
```

This code refers to the process of serializing the private key in PEM format. In this case the private key will be PKCS8 which this is a standard format for key storage. The last line of the code ensures that the private key should not be encrypted but will remain as plaintext.

RSA public key code

```
public_pem = public_key.public_bytes(

   encoding=serialization.Encoding.PEM,

   format=serialization.PublicFormat.SubjectPublicKeyInfo
```

This code refers to the process of serializing the public key in PEM format. In this case the private key will be encoding which this is a standard format for key storage. The last line of the code ensures that the public key should use the subject key info format to storage the key.

CHAPTER 7

RESULTS

Results of DES algorithm for both datasets

After the experiments we have concluded the following results. Below is a table shown the results after each method/function executed with DES algorithm by using the first Dataset which is the dataset that contains a large amount of data.

Table 7. 1 DES Algorithm Results by using Dataset 1

Process	**Encrypted Process**	**Decrypted Process**
Throughput	709153.567 bytes/sec	80581.488 bytes/sec
Time consuming	1.382 sec	2.231 sec
Memory consuming	151552 bytes	32768 bytes
Power consuming	12.057999 watts	11.56 watts
Key size	56 bits	56 bits
Security Level	Low	Low
Cost	Low	Low

Table 7. 2 DES Algorithm Results by using Dataset 2

Process	**Encrypted Process**	**Decrypted Process**
Throughput	637383.072 bytes/sec	925927.211 bytes/sec
Time consuming	1.40sec	1.35 sec
Memory consuming	28385 bytes	16384 bytes
Power consuming	19.272 watts	11.996 watts
Key size	56 bits	56 bits
Security Level	Low	Low
Cost	Low	Low

The above provided results show insights and numerical findings for the performance, cost, security level, and power and memory consumption by implementing the DES algorithm for both processes. The first parameter measured is throughput that indicated how fast the data processing method is. From results we can see that the encrypted process 709,153.567 bytes/sec is faster than decrypted process, 80,581.488 bytes/sec. If we referrer to time consuming, we can see that the encrypted process completed in 1.382 sec however the decrypted process required 2.231 seconds. If we are referring to the memory consuming process, we can see that 151552 bytes are utilizing from memory in the encrypted process and 32768 bytes for the decrypted process. Regarding the security level and cost for this algorithm seems to be low. We are analyzing a algorithm that has the key 56 bits meaning that is not very secure compared with other algorithms. The cost depends on various factors but since the security parameter is low, the key size is low and the algorithm is easy to be implemented, that means that the cost is low as well.

Regarding the second results, we are seeing that the throughput process for encrypted needs 637383 bytes/sec and for decrypts is needed 925917 bytes/sec. The encrypted process needs 28385 bytes to consume and the decrypts process needs 16384 bytes.

The below histograms and scatters plot, are very useful in order to have a better idea of the results and to make it easy to differentiate the processes. The visualization section presents the results while using the first dataset, the one that had a large number of records compared with the second one.

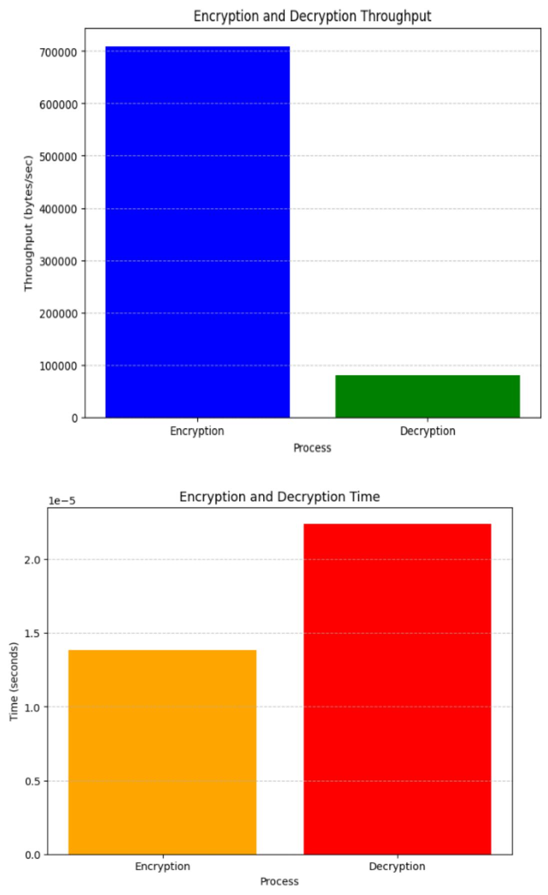
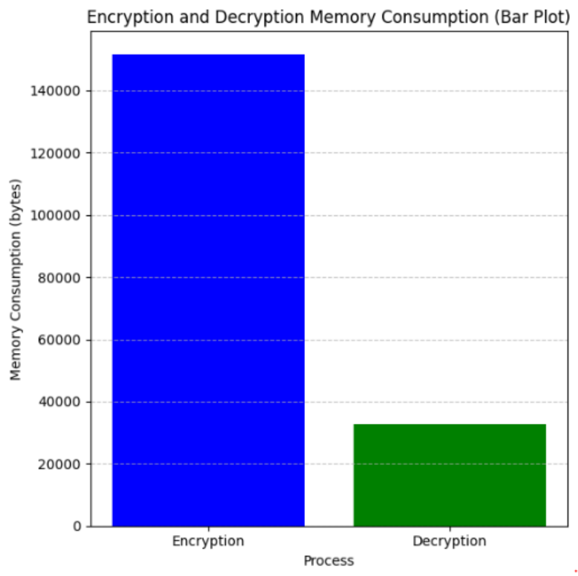


Figure 7. 1 DES Encryption and Decryption Time and Throughput



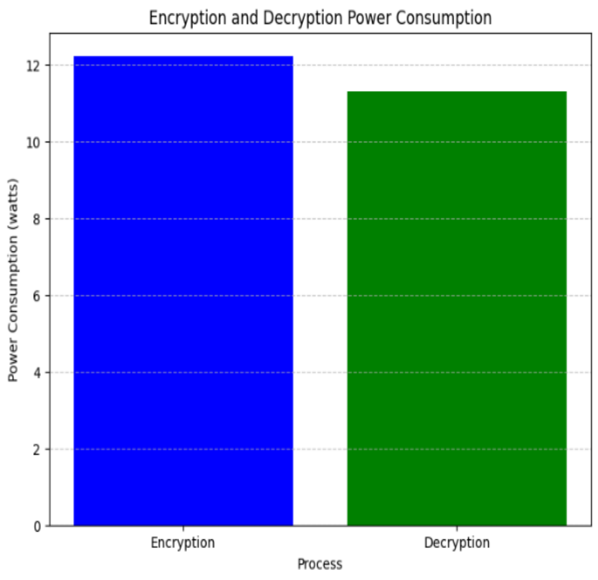
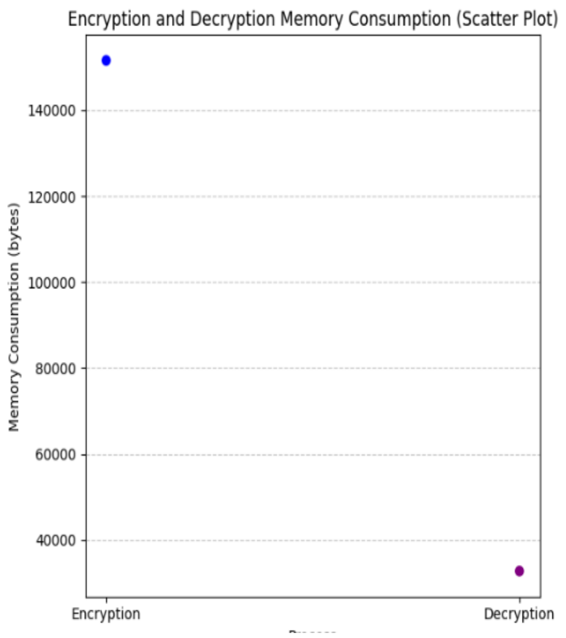


Figure 7. 2 DES Encryption and Decryption Memory and Power consumption



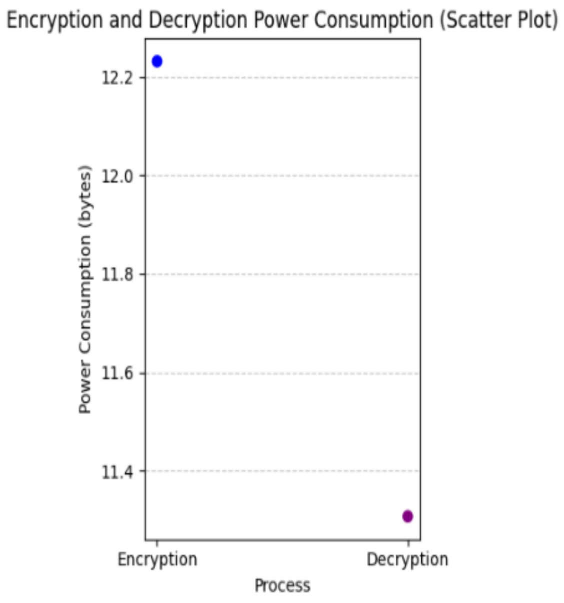


Figure 7. 3 Scatter Plot DES Encryption and Decryption Memory and Power consumption

Results of AES algorithm for both dataset

Table 7. 3 AES Algorithm Results by using Dataset 1

Process	**Encrypted Process**	**Decrypted Process**
Throughput	1051389.269bytes/sec	1032134.926 bytes/sec
Time consuming	2.8 sec	3.1 sec
Memory consuming	135663616 bytes	3497984 bytes
Power consuming	10.468 watts	14.0287 watts
Key size	128 bits	128 bits
Security Level	High	High
Cost	High	High

Table 7. 4 AES Algorithm Results by using Dataset 2

Process	**Encrypted Process**	**Decrypted Process**
Throughput	1570679.734bytes/sec	1566767.378bytes/sec
Time consuming	1.40sec	1.35 sec
Memory consuming	0 bytes	4644864 bytes
Power consuming	22.066 watts	14.88 watts
Key size	56 bits	56 bits
Security Level	Low	Low
Cost	Low	Low

The above provided results show insights and numerical findings for the performance, cost, security level, and power and memory consumption by implementing the AES algorithm for both processes. The first parameter measured is throughput that indicated how fast the data processing method is. From results we can see that the encrypted process 1051389.269bytes/sec is faster than decrypted process, 1032134.926 bytes/sec. If we referrer to time consuming, we can see that the encrypted process completed in 2.8 sec however the decrypted process required 3.1 seconds. If we are referring to the memory consuming process, we can see that 135663616 bytes are utilizing from memory in the encrypted process and 3497984 bytes for the decrypted process. Regarding the security level and cost for this algorithm seems to be high. We are analyzing a algorithm that has the key 128 bits meaning that is not very secure compared with other algorithms. The cost depends on various factors but since the security parameter is high, the key size is high and the algorithm is easy to be implemented, that means that the cost is high as well.

Regarding the second results, we are seeing that the throughput process for encrypted needs 1570679.734 bytes/sec and for decrypts is needed 1566767.378 bytes/sec.

The below histograms are very useful in order to have a better idea of the results and to make it easy to differentiate the processes. The visualization section presents the results while using the first dataset, the one that had a large number of records compared with the second one.

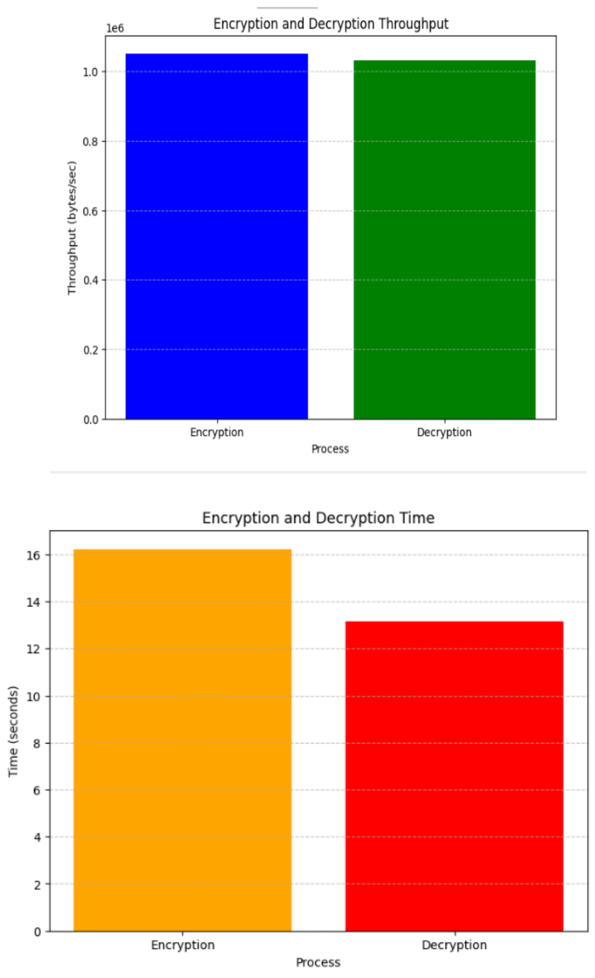


Figure 7. 4 RSA Encryption and Decryption Throughput and Time

Results of RSA algorithm for both datasets

Table 7. 5 RSA Algorithm Results by using Dataset 1

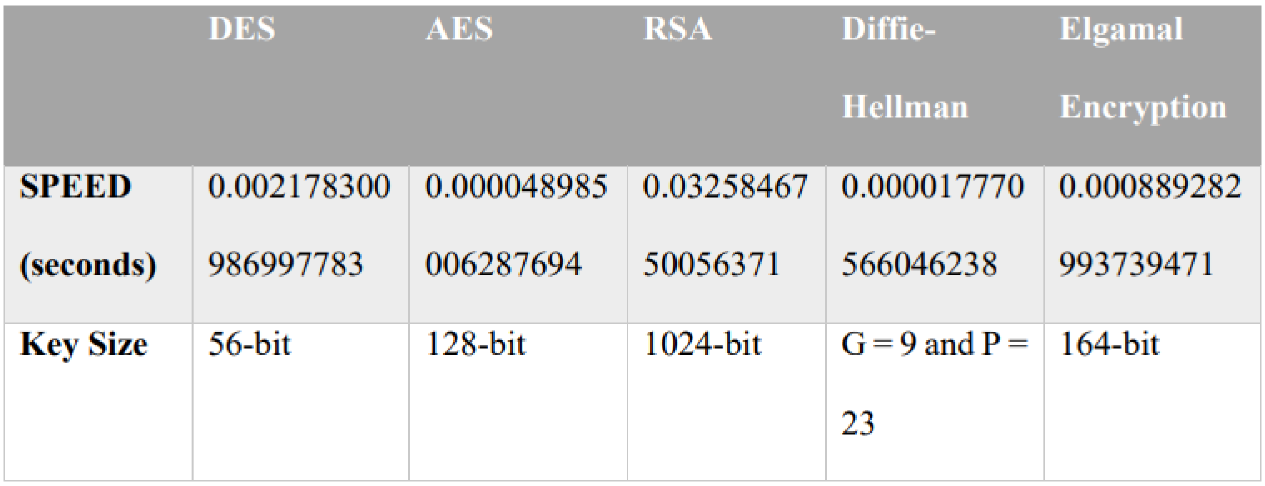
Process	**Encrypted Process**	**Decrypted Process**
Throughput	95718.269bytes/sec	379693.567 bytes/sec
Time consuming	42.800 sec	400.455 sec
Memory consuming	1461.62 MB	1517.80 MB
Power consuming	9.75 watts	8.82 watts
Key size	2048bits	2048 bits
Security Level	High	High
Cost	High	High

Table 7. 6 RSA Algorithm Results by using Dataset 2

Process	**Encrypted Process**	**Decrypted Process**
Throughput	206666.055 bytes/sec	274382.652 bytes/sec
Time consuming	6.798 sec	107.6275 sec
Memory consuming	1263.02MB	1289.3 MB
Power consuming	8.8 watts	6.55 watts
Key size	2048bits	2048 bits
Security Level	High	High
Cost	High	High

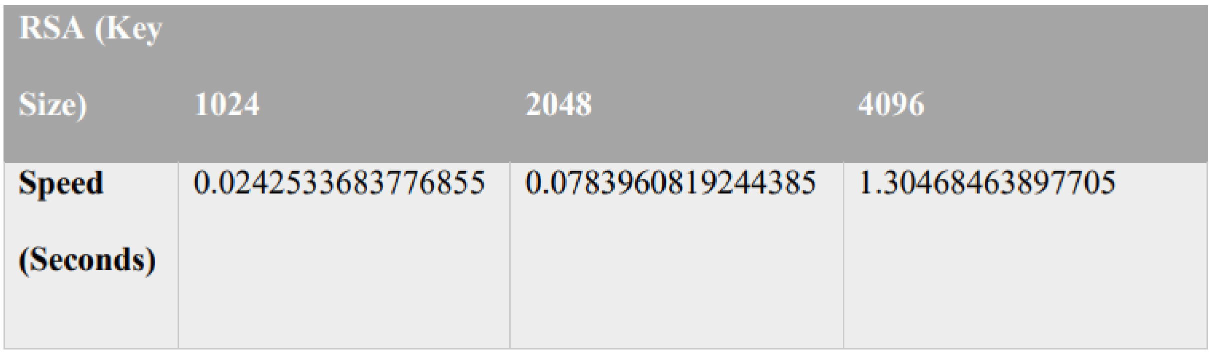
The RSA algorithm has an encrypted throughput of 95718.269bytes/sec and the time is 42 seconds. The key size of the algorithm is 2048 meaning a more time for this algorithm to execute. The security level is high since it uses two key to encrypt and decrypt data and also the cost is high. From the results we can see that executing the second dataset requires less time. The throughput of the second dataset is 206666.055 bytes/sec for the encrypted process and 274382.652 bytes/sec for decrypted process. The time to processed with the encrypted process is better that decrypted process. The key size is 2048 and since the key has a large size it makes sense to require more time to execute its functions.

Results from the Literature Review



	DES	AES	RSA	Diffie-Hellman	Elgamal Encryption
SPEED (seconds)	0.002178300986997783	0.000048985006287694	0.0325846750056371	0.000017770566046238	0.000889282993739471
Key Size	56-bit	128-bit	1024-bit	G = 9 and P = 23	164-bit

Table: Performance Comparison among Five Encryption Algorithm



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

CHAPTER 8

CONCLUSION

Based on the results generated from the experiment completed with DES, AES and RSA algorithm using two datasets, we have concluded the below:

1. **Throughput**: From results, AES demonstrates the highest throughput for processes, encrypted 1570679.734bytes/sec and decrypted 1566767.378bytes/sec across both datasets. From these results, the AES is more efficient in processing large amount of data when comparing with DES and RSA.
2. **Time consuming:** Again, the AES requires less time to generate for both processes. However the decryption process for RSA requires higher time consumption, 107.6275 sec, indicating a difference in the results between encryption and decryption process in RSA.
3. **Memory Consuming:** DES and AES algorithms shows lower memory consumption compared with RSA especially for Dataset 1. From results we can see that RSA consume more memory, especially for dataset 1, 1517.80 MB.
4. **Power Consuming:** AES is more predisposed to consume power compared with DES and RSA particularly during encryption process. RSA consume less power during decryption compared to encryption. DES shows a decrease in power consumption with dataset 2.
5. **Key Size and security level:** From the three algorithms, the most secure one is RSA with a key
6. **Cost**: From results, DES and AES algorithm has a low cost due to the low complexity and smaller key sizes. However RSA has a high cost since the key size is large.

In summary, AES is more preferred chose for application requiring high throughput and moderate security while RSA is more effective while considering security despite the high cost. DES can still be used in certain system however it is considered less secured compared with AES and RSA. Based on the parameters, the AES is an algorithm that will be used in long-terms by providing high performance.

CHAPTER 9

DISCUSSIONS

The tests conducted with the DES, AES, and RSA calculations utilizing Dataset 1 and Dataset 2 uncover nuanced trade-offs between throughput, time utilization, memory utilization, control utilization, and security level. Whereas AES illustrates high throughput and generally more time utilization, its higher control utilization compared to DES and RSA raises concerns for energy-efficient applications. RSA, in spite of advertising higher security with its bigger key measure, shows altogether higher time utilization and memory overhead, especially apparent in decoding forms and bigger datasets. These discoveries emphasize the significance of considering different variables, counting execution, security, and asset utilization, when selecting cryptographic calculations for particular applications. Moreover, the watched contrasts between Dataset 1 and Dataset 2 highlight the required for scalability considerations in cryptographic calculation determination, especially in scenarios including huge amount of information. Eventually, the choice of cryptographic calculation ought to adjust with the particular prerequisites and limitations of the application, adjusting execution, security, and asset productivity.

REFERENCES

1. A. Ochani, "DNA Image Encryption Using Modified Symmetric Key (MSK)," IEEE.
2. A_Comparative_Study_of_Twofish_Blowfish_and_Advanced_Encryption.pdf (2023).
3. Agrawal, Himani, and Monisha Sharma. , “Implementation and analysis of various symmetric cryptosystems.,” .Indian Journal of science and Technology, 3(12) pp. 1173-1176. 2010
4. Al Hasib, A., & Haque, A. A. M. M. (2008, November). A comparative study of the performance and security issues of AES and RSA cryptography.
5. Alanazi, H., Zaidan, B. B., Zaidan, A. A., Jalab, H. A., Shabbir, M., & Al-Nabhani, Y. (2010). New comparative study between DES, 3DES and AES within nine factors. arXiv preprint arXiv:1003.4085.
6. Alanazi, Hamdan, B. Bahaa Zaidan, A. Alaa Zaidan, Hamid A. Jalab, M. Shabbir, and Yahya Al-Nabhani. , “New comparative study between DES, 3DES and AES within nine factors.,” .arXiv preprint, 2010.
7. B. Kaliski, "A Survey of Encryption Standards," RSA Laboratories , 1993
8. Ebrahim, M., Khan, S., & Khalid, U. B. (2014). Symmetric algorithm survey: a comparative analysis. arXiv preprint arXiv:1405.0398.
9. Elminaam, Hatem Mohamed Abdul Kader and Mohie Mohamed Hadhoud, “Performance Evaluation of Symmetric Encryption Algorithms.” IJCSNS International Journal of Computer Science and Network Security, 8(12) pp. 280-286. 2008.
10. Farah, S., Javed, Y., Shamim, A., & Nawaz, T. (2012, December). An experimental study on performance evaluation of asymmetric encryption algorithms. In Recent Advances in Information Science, Proceeding of the 3rd European Conf. of Computer Science,(EECS-12) (pp. 121-124)
11. Fatima, A., & Nishchal, N. K. (2016). Discussion on comparative analysis and a new attack on optical asymmetric cryptosystem. JOSA A, 33(10), 2034-2040
12. Gong, Lina, Li Zhang, Wei Zhang, Xuhong Li, Xia Wang, and Wenwen Pan., “The application of data encryption technology in computer network communication security,” .In AIP Conference Proceedings .

13. Hamouda, B. E. H. H. (2020). Comparative study of different cryptographic algorithms. Journal of Information Security, 11(3), 138-148.
14. Hercigonja, Z. (2016). Comparative analysis of cryptographic algorithms. International Journal of Digital Technology & Economy, 1(2), 127-134
15. Hercigonja, Zoran., “Comparative analysis of cryptographic algorithms.,” .International Journal of Digital Technology and Economy, 1(2) pp. 127-134. 2016.
16. Hercigonja, Zoran., “Comparative analysis of cryptographic algorithms.,” .International Journal of Digital Technology and Economy, 1(2) pp. 127-134. 2016. [11] Elminaam, Diaa Salama Abd, Hatem Mohamed Abdual-Kader, and Mohiy Mohamed Hadhoud. , “Evaluating the performance of symmetric encryption algorithms.,” .IJ Network Security, 10(3) pp. 216-222. 2010.
17. Hossain, Md Alam, Md Biddut Hossain, Md Shafin Uddin, and Shariar Md Imtiaz. , “Performance analysis of different cryptography algorithms” .International Journal of Advanced Research in Computer Science and Software Engineering, 6(3) pp. 659-663. 2016
18. Hossain, Md Alam, Md Biddut Hossain, Md Shafin Uddin, and Shariar Md Imtiaz. , “Performance analysis of different cryptography algorithms” .International Journal of Advanced Research in Computer Science and Software Engineering, 6(3) pp. 659-663. 2016
19. Jeeva, A. L., Palanisamy, D. V., & Kanagaram, K. (2012). Comparative analysis of performance efficiency and security measures of some encryption algorithms. International Journal of Engineering Research and Applications (IJERA), 2(3), 3033-3037.
20. Kumar, M. Anand, and S. Karthikeyan. , “Investigating the efficiency of Blowfish and Rejindael (AES) algorithms.,” .International Journal of Computer Network and Information Security., 4(2) 2012.
21. 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.
22. M. Abdalla, M. Bellare and P. Rogawayz, "DHIES: An encryption scheme based on the Diffie-Hellman Problem," 2001

23. Mahto, D. (2017) (PDF) RSA and ECC: A comparative analysis, Research gate.
24. 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.
25. Maqsood, F., Ahmed, M., Ali, M. M., & Shah, M. A. (2017). Cryptography: a comparative analysis for modern techniques. International Journal of Advanced Computer Science and Applications, 8(6).
26. 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.
27. Mitali, V. K., & Sharma, A. (2014). A survey on various cryptography techniques. International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), 3(4), 307-312.
28. Mohammad, O. K. J., Abbas, S., El-Horbaty, E. S. M., & Salem, A. B. M. (2013, December). A comparative study between modern encryption algorithms based on cloud computing environment. In 8th International Conference for Internet Technology and Secured Transactions (ICITST-2013) (pp. 531-535). IEEE.
29. Prajapati, P., Patel, N., Macwan, R., Kachhiya, N., & Shah, P. (2014). Comparative analysis of DES
30. Rihan, Shaza D., Ahmed Khalid, and Saife Eldin F. Osman., “A performance comparison of encryption algorithms AES and DES.” .International Journal of Engineering Research and Technology (IJERT). , 4(12) pp. 151-154. 2015.
31. Rihan, Shaza D., Ahmed Khalid, and Saife Eldin F. Osman., “A performance comparison of encryption algorithms AES and DES.” .International Journal of Engineering Research and Technology (IJERT). , 4(12) pp. 151-154. 2015.
32. S. Ahmad, “A Comparison between Symmetric and Asymmetric Key Encryption Algorithm based Decryption Mixnets,” Graduate School of Engineering, University of Fukui. 2015.
33. 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
34. 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.

35. Semwal, P., & Sharma, M. K. (2017, September). Comparative study of different cryptographic algorithms for data security in cloud computing. In 2017 3rd International Conference on Advances in Computing, Communication & Automation (ICACCA)(Fall) (pp. 1-7).
36. Seth, Shashi Mehrotra, and Rajan Mishra. , “Comparative analysis of encryption algorithms for data communication 1.” 2(2) pp. 292-294,20
37. Singh, Sombir, Sunil K. Maakar, and Sudesh Kumar., “A performance analysis of DES and RSA cryptography.” .International Journal of Emerging Trends and Technology in Computer Science (IJETTCS), 2(3) pp. 418-423. 2013.
38. Singh, Sombir, Sunil K. Maakar, and Sudesh Kumar., “A performance analysis of DES and RSA cryptography.” .International Journal of Emerging Trends and Technology in Computer Science (IJETTCS), 2(3) pp. 418-423. 2013
39. Singhal, Nidhi, and J. P. S. Raina. , “Comparative analysis of AES and RC4 algorithms for better utilization.,” .International Journal of Computer Trends and Technology, 2(6) pp. 177-181. 2011.
40. Singhal, Nidhi, and J. P. S. Raina. , “Comparative analysis of AES and RC4 algorithms for better utilization.,” .International Journal of Computer Trends and Technology., 2(6) pp. 177-181. 2012.
41. 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) pp. 6-12. 2011.
42. Thambiraja, E., Ramesh, G., & Umarani, D. R. (2012). A survey on various most common encryption techniques. International journal of advanced research in computer science and software engineering, 2(7)