

Image Encryption Using RSA

Project report submitted in partial fulfillment of the requirement for the degree
of Bachelor of Technology

in

Computer Science Engineering / Information Technology

By

SALIL VERMA (191514)

SAHIL (191443)

UNDER THE SUPERVISION OF

Dr. Pankaj Dhiman

to



Department of Computer Science & Engineering and Information Technology

**Jaypee University of Information Technology Wahnaghat, Solan-173234,
Himachal Pradesh**

CERTIFICATE

Candidate's Declaration

We hereby declare that the work presented in this report entitled “ **Image Encryption using RSA**” in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering/Information Technology submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of our own work carried out over a period from January 2023 to May 2023 under the supervision of **Dr. Pankaj Dhiman (Assistant Professor SG with CSE & IT)**. I also authenticate that we have carried out the above-mentioned project work under the proficiency stream **Information Security**.

The matter embedded in the report has not been submitted for the award of any other degree or diploma.

(.....)

Sahil

(191443)

(.....)

Salil Verma

(191514)

This is to certify that the above statement made by the candidates is true to the best of my knowledge.

(.....)

Dr. Pankaj Dhiman

Assistant Professor (SG) - CSE & IT

Dated: 1/05/2023

PLAGIARISM VERIFICATION REPORT

Date: 01/05/23

Type of Document (Tick): PhD Thesis M.Tech Dissertation/ Report B.Tech Project Report Paper

Name: SALIL VERMA, SAHIL Department: CSE&IT Enrolment No 191443, 191514

Contact No. 9871014885, 8685023263 E-mail. 191514@JUITSOAN.IN, 191443@JUITSOAN

Name of the Supervisor: DR. PANKAJ DHIMAN

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

IMAGE ENCRYPTION USING RSA

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/ revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages = 70
- Total No. of Preliminary pages = 9
- Total No. of pages accommodate bibliography/references = 61

Salil Verma
(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found Similarity Index at 17.....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

[Signature]
(Signature of Guide/Supervisor)

[Signature]
(Signature of HOD)

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
Report Generated on	<ul style="list-style-type: none"> • All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String 		Word Counts	
			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature

Librarian

ORIGINALITY REPORT

17%

SIMILARITY INDEX

18%

INTERNET SOURCES

4%

PUBLICATIONS

5%

STUDENT PAPERS

PRIMARY SOURCES

1	www.fortinet.com Internet Source	1%
2	www.javatpoint.com Internet Source	1%
3	link.springer.com Internet Source	1%
4	preprod.simplilearn.com Internet Source	1%
5	www.researchgate.net Internet Source	1%
6	iosrjournals.org Internet Source	<1%
7	ijeat.org Internet Source	<1%
8	www.simplilearn.com Internet Source	1%
9	Submitted to University of Teesside Student Paper	1%

10	Submitted to University of Portsmouth Student Paper	1 %
11	ijirset.com Internet Source	1 %
12	www.tutorialspoint.com Internet Source	1 %
13	www.coursehero.com Internet Source	<1 %
14	Submitted to Jaypee University of Information Technology Student Paper	<1 %
15	Submitted to University of Wales Swansea Student Paper	<1 %
16	www.ijert.org Internet Source	<1 %
17	github.com Internet Source	<1 %
18	dockship.io Internet Source	<1 %
19	Submitted to Al Musanna College of Technology Student Paper	<1 %
20	Submitted to University of Glamorgan Student Paper	<1 %

21

docplayer.net

Internet Source

<1 %

22

obsproject.com

Internet Source

<1 %

23

pdfslide.net

Internet Source

<1 %

ACKNOWLEDGEMENT

Firstly, we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible for us to complete the project work successfully.

We are grateful and wish our profound indebtedness to Supervisor **Dr. Pankaj Dhiman**, Department of CSE & IT, Jaypee University of Information Technology, Wagnaghat. Deep Knowledge & keen interest of our supervisor in the field of “**Image Security**” to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

We would like to express our heartiest gratitude to **Dr. Pankaj Dhiman**, Department of CSE, for his kind help to finish my project. We would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, we might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, we must acknowledge with due respect the constant support and patience of our parents.

Salil Verma (191514)

Sahil (191443)

TABLE OF CONTENT

Title	Page No.
Candidate's Declaration	i
Plagiarism Certificate	ii
Acknowledgement	iii
Table of Contents	iv-v
List of Figures	vi
List of Graphs	vii
List of Tables	viii
Abstract	ix
Chapter-1 (Introduction)	
1.1 Introduction	1-8
1.2 Problem Statement	9
1.3 Objectives	10
1.4 Methodology	11
1.5 Organization	12
Chapter-2 (Literature Review)	13-15
Chapter-3 System Development	
3.1 RSA Algorithm	16
3.2 Steps of RSA Algorithm	17-19
3.3 RSA Algorithm Illustration	20-28
3.4 Proposed Algorithm	29-30
3.5 Applications of RSA Algorithm	31
3.6 Encryption	32
3.7 Decryption	33

Chapter-4 Performance Analysis	34
4.1 Practical Work	35-45
Chapter-5 Conclusions	
5.1 Conclusions	46
5.2 Recommendations	46
5.3 Applications	47-49
References	50
Appendices	51-61

LIST OF FIGURES

Figure no.	Figure Info.
Figure 1	Symmetric cryptography
Figure 2	Asymmetric cryptography
Figure 3	CIA Triad
Figure 4	Original Image
Figure 5	Encrypted Image
Figure 6	Methodology of RSA
Figure 7	Encryption Decryption
Figure 8	Plain text to Cipher text
Figure 9	Encyrption
Figure 10	Decryption
Figure 11	The image used for encryption
Figure 12-20	Implementing Code
Figure 21	Encrypted Image
Figure 22	Decrypted Image

LIST OF GRAPHS

Graph no.	Graph Info.
Graph 1	Graph showing effect of data input on CPU-RSA and GPU-RSA along with the Speedup
Graph 2	Flowchart-of-RSA-encryption-and-decryption-operations
Graph 3	Graph showing various encryption methods with time

LIST OF TABLES

Table no.	Table Info.
Table 1	Time established to generate two prime numbers
Table 2	Observation Table 1:(P, Q, E, D, Time)
Table 3	Observation Table 2:(P, Q, E, D, Time)
Table 4	Observation Table 3:(P, Q, E, D, Time)
Table 5	Observation Table 4:(P, Q, E, D, Time)
Table 6	Observation Table 5:(P, Q, E, D, Time)

ABSTRACT

In today's world, secure online data transmission takes precedence over other activities. Various algorithms exist to provide the computational difficulty that makes cracking the key to identify a unique message difficult. Many researchers have applied various cryptographic algorithms for secure data transmission, and various hybrid cryptographic algorithms have been proposed to improve the level of information security. Key management plays an important role in implementing cryptographic algorithms. For this reason, we applied image encryption technology that uses a random image as a key. I used a random image as a key and encrypted another image as information using the RSA algorithm. The proposed method is compared with conventional approaches and concludes that encryption algorithms implemented using images as keys offer more security in terms of encryption and decryption times.

Chapter 01: INTRODUCTION

1.1 Introduction

Information is presented in a variety of formats, including text, images, audio, and video. Security and authentication are becoming increasingly important when transferring data over the Internet. Therefore, different researchers use large keys while implementing different encryption algorithms to increase the level of data security. The larger the key, the more sensitive the data, but the management of the key becomes a tedious task. Using images as keys instead of large format keys makes key management easier and improves information security. Encryption technology is widely used to achieve information security. Image encryption is the process of transforming the actual image into a different, secure format. As a result, the information is relatively difficult to access and destroy without the decryption key. RSA is one of the recognized public-key cryptographic algorithms that facilitates encryption and authentication of information.

There are different uses for image encryption and decryption in digital communications. Image security is often used in other areas such as medical imaging, military information multimedia structures, and telemedicine. In hospitals, finance, government, defence and other sectors, images are used as keys to provide a higher level of data protection than information is the key. Stealing personal information is an illegal duty for any kind of institution. Therefore, stronger methods of encrypting images or data are required to ensure that data is transmitted securely and cannot be easily hacked by attackers. On the receiving side, you can find the actual image after decoding. The receiving end can easily access the data or image using the private key.

Basic Terms Used in Cryptography

Plain Text: The original message that the person wants to convey to the other party is defined as plain text. In encryption, the actual message that needs to be sent to the remote station is specially named as plain text.

Cipher Text: A meaningless message that no one can understand is called a cipher text. Encryption converts the original message into a readable message before sending the actual message.

Encryption: The process of converting plain text into cipher text is called encryption. Encryption uses cryptographic techniques to send confidential messages over insecure channels. The encryption process requires two things: an encryption algorithm and a key. Cryptographic Algorithm means the technology used for encryption. Encryption is done at the sender.

Decryption: The reverse encryption process is called decryption. This is the process of converting ciphertext to plaintext. In encryption, the receiver uses decryption techniques to retrieve the original message from a readable message (the ciphertext). The decryption process requires two things: a decryption algorithm and a key. A decryption algorithm describes the technique used for decryption. In general, encryption and decryption algorithms are the same.

Key: Keys can be numeric or alphanumeric text, or special symbols. Keys are used when plaintext is encrypted and ciphertext is decrypted. The choice of keys in cryptography is very important because the security of encryption algorithms directly depends on the keys.

Purpose of Cryptography

Encryption offers many security goals to ensure privacy, tamper-proofing of data, etc. It is widely used today due to the great security benefits of encryption. Below are the different goals of encryption.

Confidentiality: Information on your computer is transmitted and should only be retrieved by authorized parties and not by others.

Authentication: Information received by each system must verify the identity of the sender, whether that information is from an authorized person or a false identity.

Integrity: Only authorized parties may change submitted information. A given message cannot be changed between sender and recipient.

Non-Repudiation: Prevents senders and recipients of messages from refusing forwarding.

Access Control: Only authorized parties can access the information provided.

Classification of Cryptography

Cryptography technique is used when secret messages are transferred from one party to another over a communication line.

There are two main types of cryptography:

1. Symmetric key cryptography

In this type of encryption, both sender and receiver know the same secret code called key. Messages are encrypted with a key by the sender and decrypted with the same key by the recipient. Keys play a very important role in symmetric cryptography. This is because its security directly depends on the type of key, i.e., key length etc.

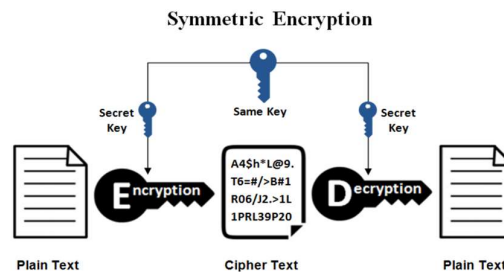


Fig 1:Symmetric Encryption

2. Asymmetric key cryptography

Asymmetric key cryptography is used as a pair of encryption and decryption algorithms. In public-key cryptography, a key works with a corresponding public-private key pair.

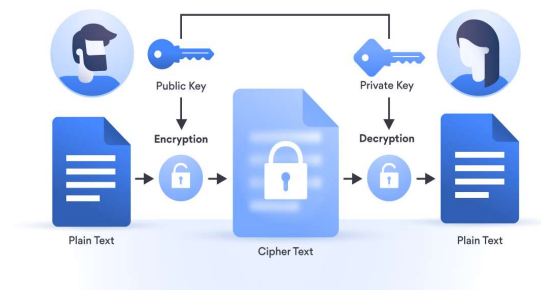


Fig 2:Asymmetric Encryption

What is the CIA Triad?

The three letters in "CIA triad" stand for **Confidentiality, Integrity, and Availability**. The CIA triad is a common model that forms the basis for the development of security systems. They are used for finding vulnerabilities and methods for creating solutions.

The confidentiality, integrity, and availability of information is crucial to the operation of a business, and the CIA triad segments these three ideas into separate focal points. This differentiation is helpful because it helps guide security teams as they pinpoint the different ways in which they can address each concern.

Ideally, when all three standards have been met, the security profile of the organization is stronger and better equipped to handle threat incidents.



Fig 3: CIA Triad

Confidentiality

Confidentiality involves the efforts of an organization to make sure data is kept secret or private. To accomplish this, access to information must be controlled to prevent the unauthorized sharing of data—whether intentional or accidental. A key component of maintaining confidentiality is making sure that people without proper authorization are prevented from accessing assets important to your business. Conversely, an effective system also ensures that those who need to have access have the necessary privileges.

For example, those who work with an organization's finances should be able to access spreadsheets, bank accounts, and other information related to the flow of money. However, most other employees—and perhaps even certain executives—may not be granted access. To ensure these policies are followed, stringent restrictions must be in place to limit who can see what.

There are several ways confidentiality can be compromised. This may involve direct attacks aimed at gaining access to systems the attacker does not have the right to see. It can also involve an attacker making a direct attempt to infiltrate an application or database so they can take data or alter it.

These direct attacks may use techniques such as man-in-the-middle (MITM) attacks, where an attacker positions themselves in the stream of information to intercept data and then either steal or alter it. Some attackers engage in other types of network spying to gain access to credentials. In some cases, the attacker will try to gain more system privileges to obtain the next level of clearance.

However, not all violations of confidentiality are intentional. Human error or insufficient security controls may be to blame as well. For example, someone may fail to protect their password—either to a workstation or to log in to a restricted area. Users may share their credentials with someone else, or they may allow someone to see their login while they enter it. In other situations, a user may not properly encrypt a communication, allowing an attacker to intercept their information. Also, a thief may steal hardware, whether an entire computer or a device used in the login process and use it to access confidential information.

Integrity

Integrity involves making sure your data is trustworthy and free from tampering. The integrity of your data is maintained only if the data is authentic, accurate, and reliable.

For example, if your company provides information about senior managers on your website, this information needs to have integrity. If it is inaccurate, those visiting the website for information may feel your organization is not trustworthy. Someone with a vested interest in damaging the reputation of your organization may try to hack your website and alter the descriptions, photographs, or titles of the executives to hurt their reputation or that of the company.

Compromising integrity is often done intentionally. An attacker may bypass an intrusion detection system (IDS), change file configurations to allow unauthorized access, or alter the logs kept by the system to hide the attack. Integrity may also be violated by accident. Someone may accidentally enter the wrong code or make another kind of careless mistake. Also, if the company's security policies, protections, and procedures are inadequate, integrity can be violated without any one person in the organization accountable for the blame.

To protect the integrity of your data, you can use hashing, encryption, digital certificates, or digital signatures. For websites, you can employ trustworthy certificate authorities that verify the authenticity of your website, so visitors know they are getting the site they intended to visit.

A method for verifying integrity is non-repudiation, which refers to when something cannot be repudiated or denied. For example, if employees in your company use digital signatures when sending emails, the fact that the email came from them cannot be denied. Also, the recipient cannot deny that they received the email from the sender.

Availability

Even if data is kept confidential and its integrity maintained, it is often useless unless it is available to those in the organization and the customers they serve. This means that systems, networks, and applications must be functioning as they should and when they should. Also, individuals with access to specific information must be able to consume it when they need to, and getting to the data should not take an inordinate amount of time.

If, for example, there is a power outage and there is no disaster recovery system in place to help users regain access to critical systems, availability will be compromised. Also, a natural disaster like a flood or even a severe snowstorm may prevent users from getting to the office, which can interrupt the availability of their workstations and other devices that provide business-critical information or applications. Availability can also be compromised through deliberate acts of sabotage, such as the use of denial-of-service (DoS) attacks or ransomware.

To ensure availability, organizations can use redundant networks, servers, and applications. These can be programmed to become available when the primary system has been disrupted or broken. You can also enhance availability by staying on top of upgrades to software packages and security systems. In this way, you make it less likely for an application to malfunction or for a relatively new threat to infiltrate your system. Backups and full disaster recovery plans also help a company regain availability soon after a negative event.

1.2 Problem Statement

Modern age with advanced technology. Most people prefer to use the internet as their primary means of transferring data from one end to the other over the internet. There are many ways to send data over the Internet, including: Send emails, texts and images. Images are widely used in the modern world of telecommunications. However, one of the main concerns when sending data over the Internet is "security" and reliability. Data security basically means protecting data from unauthorized users and intruders. Encryption is a method of protecting information. Image coding is the technique of transforming an original image into another, less comprehensible form. No one can access the content without knowing the decryption key.

1.3 Objectives

Digital images are widely disseminated on the Internet. With the development of Internet technology, digital images containing a lot of visual information are transmitted and stored over the Internet and can be copied, manipulated, and illegally used by unauthorized users, especially in the military, commercial, and medical fields. and image security incidents can occur. Many governments, businesses and individuals are beginning to pay attention to privacy issues. Therefore, protecting your images is very important. Digital image security is an essential and difficult task over common communication channels. Various techniques are used to protect digital images, including encryption, steganography, and watermarking. These are ways to protect digital images to meet your security goals. Confidentiality, Integrity and Availability (CIA). The RSA algorithm is the foundation of cryptosystems (a set of cryptographic algorithms used for a particular security service or purpose), enables public-key cryptography, and is widely used to protect sensitive data. such as the Internet. The purpose of image encryption schemes is to obtain the highest quality hidden image to keep the information secret.

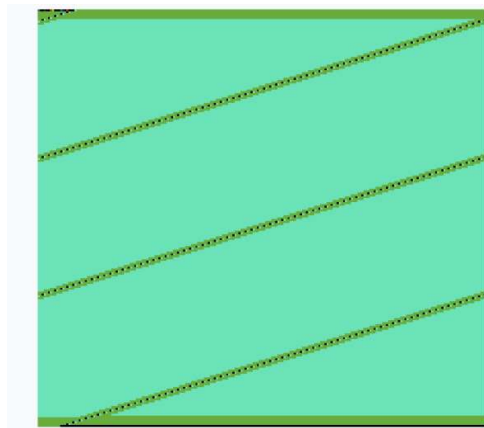


Fig 4:Original Image

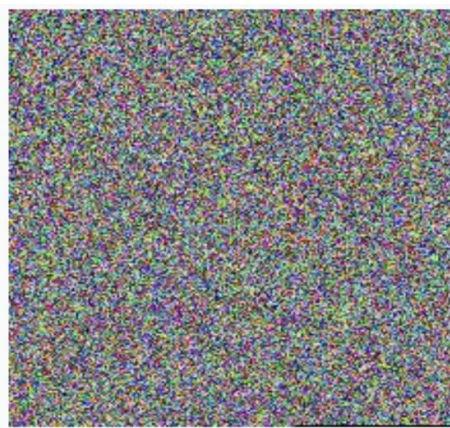


Fig 5:Encrypted Image

1.4 Methodology

RSA is a cryptographic algorithm used to encrypt and decrypt data. This algorithm was developed by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977. The RSA cryptosystem is also known as the public key cryptosystem. RSA is commonly used for secure data transfer. According to the RSA algorithm, encryption starts with choosing two large prime numbers and an auxiliary value as the public key. Prime numbers are kept secret. The public key is used to encrypt messages and the private key is used to decrypt messages or information. The RSA algorithm encrypts the original image and uses a different key to decrypt the image. This is illustrated in the following diagram.

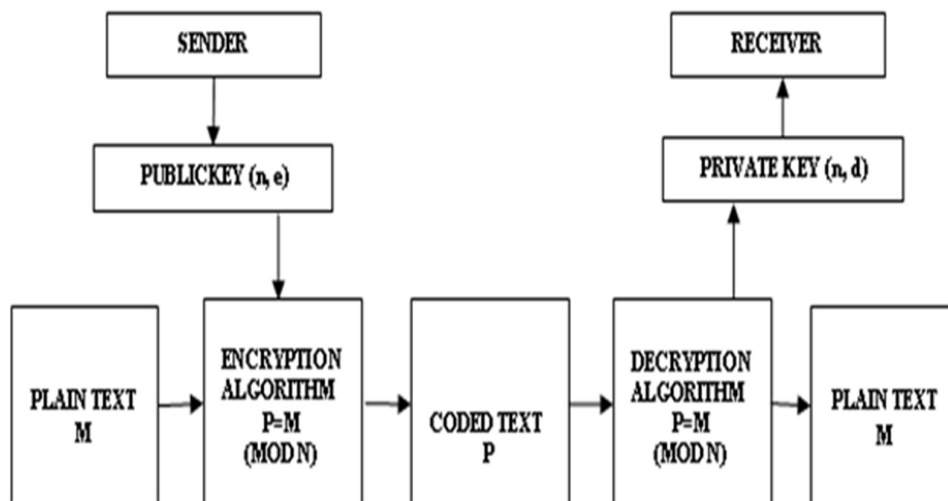


Fig 6: Methodology of RSA

1.5 Organization

Chapter 1: Introduction

Covers the various project related things such as, introduction, Problem statement, motivation and tells us about the reasons behind the choice of this project.

Chapter 2: Literature Survey

Covers the literature surveyed as well as talks about the concepts that have been studied and understood.

Chapter 3: System Development

Covers the tools and technologies that are used. It also talks about System Design- various design diagrams.

Chapter 4: Performance analysis

It covers the implementation of the project and the project snapshots.

Chapter 5: Conclusion

Chapter 02: LITERATURE SURVEY

Information security is a serious issue for modern online applications. When data is transmitted via online media, it must be protected against unauthorized access. Many researchers have proposed many security algorithms and hybrid approaches to information security. Faster algorithms and high-performance computing techniques provide ways to decrypt data even after security algorithms are implemented. Therefore, various researchers are interested in image security. Analyzing network traffic involves managing various resources where bandwidth plays a critical role. Researchers have proposed a dynamic bandwidth management technique for on-demand services. Since images require more bandwidth than textual information, the proposed operations can be integrated with dynamic resource management technology to provide faster and more secure services. Today, mixed media data travels extensively across the web in various structures such as images, audio, video, and text. Computer communication via the Internet makes all information transparent and available to all customers. Data security is therefore an important and critical task. As data moves from one place to another over the Internet, it must be protected from unauthorized access. So, the most important thing is to transfer your photos safely. There are multiple ways to store images, including encryption, watermarking, custom watermarking, cryptanalysis, and steganography.

A. A New Image Encryption Technique Based on Combination of Block Displacement and Block Cipher Technique 2013.

In this article, we propose a new image encryption algorithm. Algorithm security is already known to depend on key length. That is, longer key lengths always support better security features, and the proposed algorithm uses a key length of 128 bits. proposed algorithm. It would take him 2128 times to crack the key to access the cryptanalysis of the original or proposed key, which is almost impossible for a hacker. The proposed algorithm does not apply such kind of formulas, so floating point errors are not possible. Correlation coefficients and entropy values of the proposed algorithm were calculated.

B. Permutation based Image Encryption Technique, 2011

Sesha Pallavi Indrakanti and P.S.Avadhani introduced an algorithm based on random pixel replacement with the motivation of preserving image quality. He had three stages in the encryption process. Phase 1 was image encryption. Phase 2 was the key generation phase. Phase 3 was the identification process. This ensures confidentiality of color images with less computation.

C. Image Encryption Based on Explosive Inter-pixel Displacement of the RGB Attributes of a Pixel, 2011

Proposed in 2011, this paper focused on manipulating the RGB values of pixels and shifting them according to predefined keys. A circular shift is applied to the three components of a pixel using different keys so that the R, G, and B values of one pixel are blended with the R, G, and B values of another pixel.

D. Image Encryption based on the RGB PIXEL Transposition and Shuffling 2013.

In this paper, a method of gradually transposing and remixing the RGB values of an image was proposed, which proved to be very effective in terms of security analysis. His additional swapping of RGB values in the image file after the RGB component shift made the image more secure against all currently available attacks.

Chapter 03: SYSTEM DEVELOPMENT

3.1 RSA Algorithm

Developed by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977, the RSA algorithm is one of the most widely used asymmetric key encryption algorithms. Naming is based on the last name of the developer. This algorithm enables both information security and authentication.

Public key cryptosystems are primarily used to protect data in transit. RSA is an asymmetric key encryption algorithm that uses a public/private key pair. The sender encrypts the message with the public key and the recipient decrypts the message with the recipient's private key.

The reason for using the RSA algorithm is the main problem with factoring large integers. Security comes from the product of two large prime numbers in implementing the algorithm. The most complicated part of RSA encryption is the generation of public and private keys. Two prime numbers, p and q , are generated using the Rabin-Miller primality test algorithm. The connection between private and public keys is established by two prime numbers. Key sizes are often expressed in bits.

3.2 Steps of RSA Algorithm

The RSA algorithm involves three major steps during implementation. They are as follows:

- A. Generation of key
- B. Encryption
- C. Decryption.

A. Key Generation

The first stage of the RSA algorithm is key generation, which includes public and private generation. As the name suggests, the public key is the public key that is visible to everyone and is used to participate in encrypting messages. Image transmissions are encrypted with your public key and can be decrypted with your private key. A key for the RSA algorithm can be created using the following steps:

1. First, select the two different prime numbers that are p and q .
2. For safety, prime integers p and q should be selected with the same bit-length. Prime integers are efficiently found by primality testing.
3. Then, calculate the value of n that is $n=pq$.
4. n is the modulus that is used for equally the public and private keys. Its length is known as key length that is usually stated in bits.

5. Then compute Euler's totient of n .

$$\phi(n) = \phi(p)\phi(q) = (p-1)(q-1) = n - (p+q-1);$$
 here, ϕ is Euler's totient function. This rate is kept private

6. Then, choose an integer e that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$;
 i.e., e and $\phi(n)$ are coprime.
 e is out as a key which is kept public.
 e has a brief bit-length and slight Hamming weight outcomes
 in more effective encryption. However, minor e values have
 been published to become less locked in some settings.

7. Determine d as $d \equiv e^{-1} \pmod{\phi(n)}$,
 i.e., d which is the modular multiplicative inverse of e
 (modulo $\phi(n)$). This is performed as, solve d given by
 $d \cdot e \equiv 1 \pmod{\phi(n)}$.

That is calculated using an extended Euclidean algorithm. It uses the pseudo-code in the modular integers section; inputs a and n correspond to e and $\phi(n)$, respectively.

Evaluate the value of d which is kept as the private key. The public key involves the modulus of n and e . The private key has the modulus of n and d , and it is kept secret. p, q , and $\phi(n)$ values are kept secret because these values can be used for calculating d .

B. Encryption

$$c \equiv me \pmod{n}$$

c=cipher text

m=plain text

e=public key

d=private key

C. Decryption

$$m \equiv cn \pmod{n}$$

c=cipher text

m=plain text

e=public key

d=private key

3.3 RSA Algorithm Illustration

This method is the traditional method used for encryption and decryption using text or numeric keys. Here I proposed the same algorithm as the previous one with the simple modification of using images as keys instead of text or numbers as keys.

The following is our proposed algorithm:

Let's take some examples of RSA encryption algorithm:

Example 1:

This example shows how we can encrypt plaintext 9 using the RSA public-key encryption algorithm. This example uses prime numbers 7 and 11 to generate the public and private keys.

Explanation:

Step 1: Select two large prime numbers, p , and q .

$$p = 7$$

$$q = 11$$

Step 2: Multiply these numbers to find $n = p \times q$, where n is called the modulus for encryption and decryption.

First, we calculate $n = p \times q$

$$n = 77$$

Step 3: Choose a number e less than n , such that n is relatively prime to $(p - 1) \times (q - 1)$. It means that e and $(p - 1) \times (q - 1)$ have no common factor except 1. Choose " e " such that $1 < e < \phi(n)$, e is prime to $\phi(n)$, $\gcd(e, \phi(n)) = 1$.

Second, we calculate

$$\phi(n) = (p - 1) \times (q - 1)$$

$$\phi(n) = (7 - 1) \times (11 - 1)$$

$$\phi(n) = 6 \times 10$$

$$\phi(n) = 60$$

Let us now choose the relative prime e of 60 as 7.

Thus, the public key is $\langle e, n \rangle = (7, 77)$

Step 4: A plaintext message m is encrypted using public key $\langle e, n \rangle$. To find ciphertext from the plain text following formula is used to get ciphertext C .

To find ciphertext from the plain text following formula is used to get ciphertext C .

$$C = m^e \bmod n$$

$$C = 97 \bmod 77$$

$$C = 37$$

Step 5: The private key is $\langle d, n \rangle$. To determine the private key, we use the following formula d such that:

$$d \bmod \{(p - 1) \times (q - 1)\} = 1$$

$$7d \bmod 60 = 1, \text{ which gives } d = 43$$

The private key is $\langle d, n \rangle = (43, 77)$

Step 6: A ciphertext message c is decrypted using the private key $\langle d, n \rangle$. To calculate plain text m from the ciphertext c the following formula is used to get plain text m .

$$m = cd \bmod n$$

$$m = 3743 \bmod 77$$

$$m = 9$$

In this example, Plain text = 9 and the ciphertext = 37

Example 2:

In an RSA cryptosystem, a particular A uses two prime numbers, 13 and 17, to generate the public and private keys. If the public of A is 35. Then the private key of A is

Explanation:

Step 1: in the first step, select two large prime numbers, p and q.

$$p = 13$$

$$q = 17$$

Step 2: Multiply these numbers to find $n = p \times q$, where n is called the modulus for encryption and decryption.

First, we calculate

$$n = p \times q$$

$$n = 13 \times 17$$

$$n = 221$$

Step 3: Choose a number e less than n, such that n is relatively prime to $(p - 1) \times (q - 1)$. It means that e and $(p - 1) \times (q - 1)$ have no common factor except 1. Choose "e" such that $1 < e < \phi(n)$, e is prime to $\phi(n)$, $\gcd(e, \phi(n)) = 1$.

Second, we calculate

$$\phi(n) = (p - 1) \times (q - 1)$$

$$\phi(n) = (13 - 1) \times (17 - 1)$$

$$\phi(n) = 12 \times 16$$

$$\varphi(n) = 192$$

$$\text{g.c.d}(35, 192) = 1$$

Step 3: To determine the private key, we use the following formula to calculate the d such that:

$$\text{Calculate } d = de \bmod \varphi(n) = 1$$

$$d = d \times 35 \bmod 192 = 1$$

$$d = (1 + k \cdot \varphi(n)) / e \quad [\text{let } k = 0, 1, 2, 3, \dots, \dots]$$

Put $k = 0$

$$d = (1 + 0 \times 192) / 35$$

$$d = 1/35$$

Put $k = 1$

$$d = (1 + 1 \times 192) / 35$$

$$d = 193/35$$

Put $k = 2$

$$d = (1 + 2 \times 192) / 35$$

$$d = 11$$

The private key is $\langle d, n \rangle = (11, 221)$

Hence, private key i.e., $d = 11$

Example 3:

An RSA cryptosystem uses two prime numbers 3 and 13 to generate the public key = 3 and the private key = 7. What is the value of cipher text for a plain text?

Explanation:

Step 1: In the first step, select two large prime numbers, p and q .

$$p = 3$$

$$q = 13$$

Step 2: Multiply these numbers to find $n = p \times q$, where n is called the modulus for encryption and decryption.

First, we calculate

$$n = p \times q$$

$$n = 3 \times 13 = 39$$

Step 3: If $n = p \times q$, then the public key is $\langle e, n \rangle$. A plaintext message m is encrypted using public key $\langle e, n \rangle$. Thus, the public key is $\langle e, n \rangle = (3, 39)$.

To find ciphertext from the plain text following formula is used to get ciphertext C .

$$C = m^e \text{ mod } n$$

$$C = 5^3 \text{ mod } 39$$

$$C = 125 \text{ mod } 39$$

$$C = 8$$

Hence, the ciphertext generated from plain text, $C = 8$.

Example 4:

An RSA cryptosystem uses two prime numbers, 3 and 11, to generate private key = 7. What is the value of ciphertext for a plain text 5 using the RSA public-key encryption algorithm?

Explanation:

Step 1: in the first step, select two large prime numbers, p and q .

$$p = 3$$

$$q = 11$$

Step 2: Multiply these numbers to find $n = p \times q$, where n is called the modulus for encryption and decryption.

First, we calculate

$$n = p \times q$$

$$n = 3 \times 11$$

$$n = 33$$

Step 3: Choose a number e less than n , such that n is relatively prime to $(p - 1) \times (q - 1)$. It means that e and $(p - 1) \times (q - 1)$ have no common factor except 1. Choose " e " such that $1 < e < \varphi(n)$, e is prime to $\varphi(n)$, $\gcd(e, \varphi(n)) = 1$.

Second, we calculate

$$\varphi(n) = (p - 1) \times (q - 1)$$

$$\varphi(n) = (3 - 1) \times (11 - 1)$$

$$\varphi(n) = 2 \times 10$$

$$\varphi(n) = 20$$

Step 4: To determine the public key, we use the following formula to calculate the d such that:

Calculate $e \times d = 1 \pmod{\phi(n)}$

$$e \times 7 = 1 \pmod{20}$$

$$e \times 7 = 1 \pmod{20}$$

$$e = (1 + k \cdot \phi(n)) / d \quad [\text{let } k = 0, 1, 2, 3, \dots]$$

Put $k = 0$

$$e = (1 + 0 \times 20) / 7$$

$$e = 1/7$$

Put $k = 1$

$$e = (1 + 1 \times 20) / 7$$

$$e = 21/7$$

$$e = 3$$

The public key is $\langle e, n \rangle = (3, 33)$

Hence, public key i.e., $e = 3$

3.4 Proposed Algorithm

The proposed algorithm considered an image as information rather than text or numbers, and another image as a key to implement the RSA algorithm.

The steps of the proposed algorithm are as follows:

1. Consider the image and find out the array format.
2. Finding the length of an array.
3. Consider the key image and find out the corresponding array format.
4. From the key image, any random prime number will be considered as key (For simplicity, we have considered the first prime number as key among the key image array).
5. Then, the same key generation, encryption, decryption process can be done as the traditional one.

3.4.1 Key generation

1. Choose two numbers p and q that are prime and distinct in nature.
2. For high security purposes, the prime numbers p and q should be randomly taken and must have the same bit-length.
3. Compute $n=pq$, where n is used for modulus of both the public and private keys. Its size is expressed in bits which is known as key length.
4. Compute $\phi(n)=\phi(p)\phi(q)=(p-1)(q-1)=n-(p+q-1)$, where ϕ is Euler's totient function.
5. Choose an integer e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$, i.e., e and $\phi(n)$ are coprime.
6. Determine $d \equiv e^{-1} \pmod{\phi(n)}$;
i.e., d is the multiplicative inverse of e (modulo $\phi(n)$).

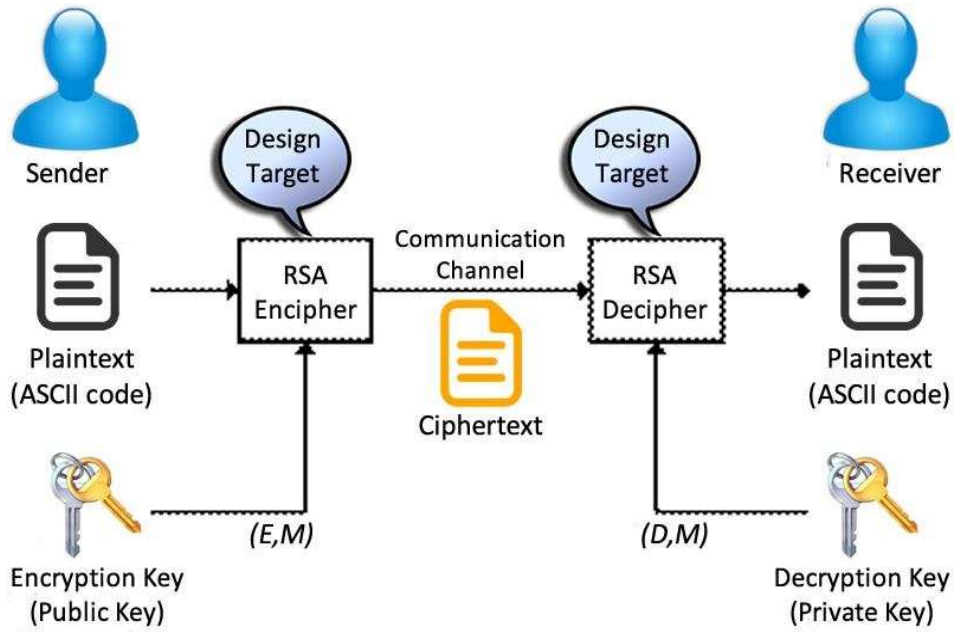


Fig 7: Encryption Decryption

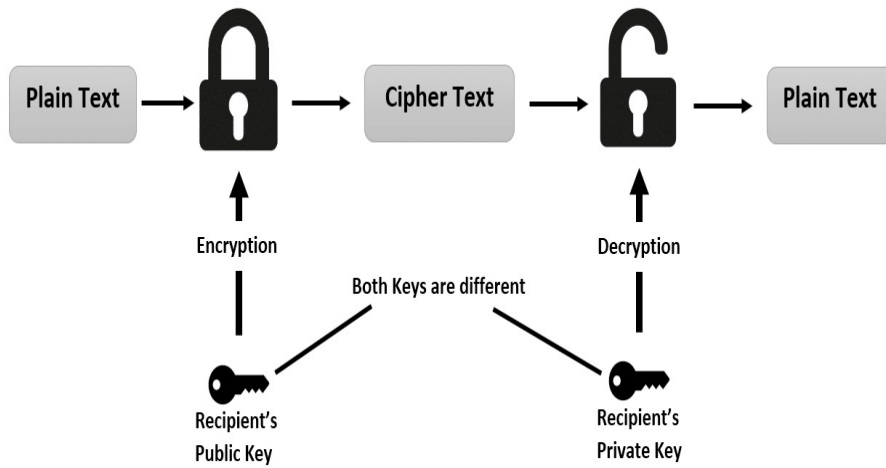


Fig 8: Plain text to Cipher text

3.5 Advantages of RSA Algorithm

- No Key Sharing: RSA encryption depends on using the receiver's public key, so you don't have to share any secret key to receive messages from others.
- Proof of Authenticity: Since the key pairs are related to each other, a receiver can't intercept the message since they won't have the correct private key to decrypt the information.
- Faster Encryption: The encryption process is faster than that of the DSA algorithm.
- Data Can't Be Modified: Data will be tamper-proof in transit since meddling with the data will alter the usage of the keys. And the private key won't be able to decrypt the information, hence alerting the receiver of manipulation.

3.6 Encryption

In encryption, the plain text content represented in the form of an image is converted to cypher text using another image as a secret key. The picture can likewise be changed over to scrambled structure utilizing the random image as key and the resultant image after encryption. The scrambled picture is then sent over an insecure channel to the receiver. At the receiver end, the scrambled picture is decoded using the private key of the receiver. The proposed image encryption technique is used for providing better security of information. Subsequent, to encoding information, objective-scrambled information was decoded with assistance of association called as unscrambling. The resultant encrypted image is generated by applying the RSA algorithm between information image and the key image i.e., the binary equivalent of information image undergoes encryption with each pixel corresponding binary equivalent of key image to produce the encrypted image.



Fig 9:Encryption

3.7 Decryption

Decryption is the process of transforming data that has been rendered unreadable through encryption back to its original form. In decryption, the system extracts and converts the garbled data and transforms it to texts and images that are easily understandable not only by the reader but also by the system. Decryption may be accomplished manually or automatically. It may also be performed with a set of keys or passwords.



Fig 10:Decryption

Chapter 04: PERFORMANCE ANALYSIS

- Choose two numbers p and q that are prime and distinct in nature.
- For high security purposes, the prime numbers p and q should be randomly taken and must have the same bit-length.
- Compute $n=pq$, where n is used for modulus of both the public and private keys. Its size is expressed in bits which is known as key length.
- Compute $\phi(n)=\phi(p)\phi(q)=(p-1)(q-1)=n-(p+q-1)$, where ϕ is Euler's totient function.
- Choose an integer e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$, i.e., e and $\phi(n)$ are coprime.
- Determine d as $d \equiv e^{-1} \pmod{\phi(n)}$;
- i.e., d is the multiplicative inverse of e (modulo $\phi(n)$).
- Solve d given $d \cdot e \equiv 1 \pmod{\phi(n)}$.

4.1 Practical Work

As mentioned earlier, there are two different primes P and Q used to generate n. Write code to generate them. As the length increases, it takes longer to generate the relative primes e and d. The following table shows the relationship between the numbers chosen for P and Q and the time allotted to generate them.

Table 1: Prime Number Time Establishment

Chosen Numbers		Time Established (seconds)
P	Q	
7	5	0.010387
11	13	0.040969
17	23	0.124453
29	53	0.964863
47	59	3.504031
113	71	18.675820
239	173	590.455964

Now we use digital images as data to be encrypted. We used an image with size 612 X 612.

PC specifications are:

System: Windows 10 Home Single Language

Processor: Intel(R) Core (TM) i3-6100U CPU @ 2.30GHz 2.30 GHz

RAM: 4 GB

System Type: 64-bit operating system, x64-based processor

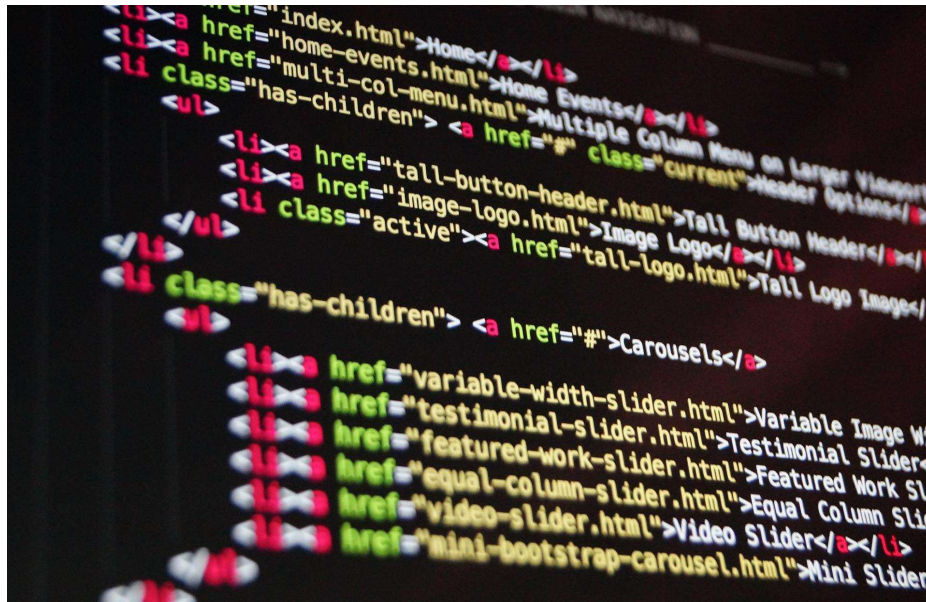
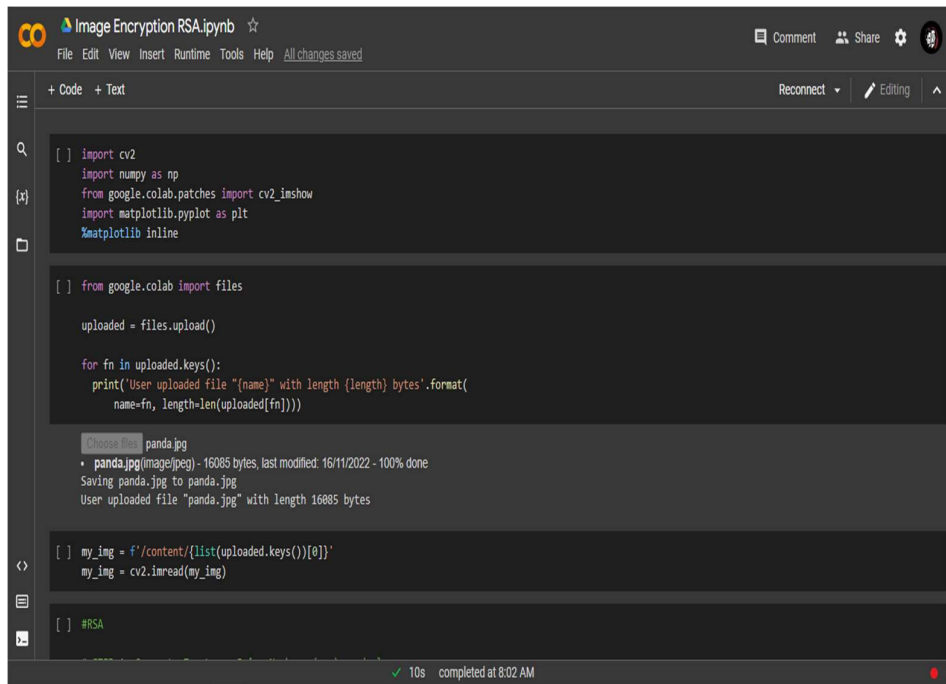


Fig 11: The image used for encryption

Screenshots Of Implementation



```
Image Encryption RSA.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text Reconnect Editing

[ ] import cv2
import numpy as np
from google.colab.patches import cv2_inshow
import matplotlib.pyplot as plt
%matplotlib inline

[ ] from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
    print('User uploaded file "{}" with length {} bytes'.format(
        name=fn, length=len(uploaded[fn])))

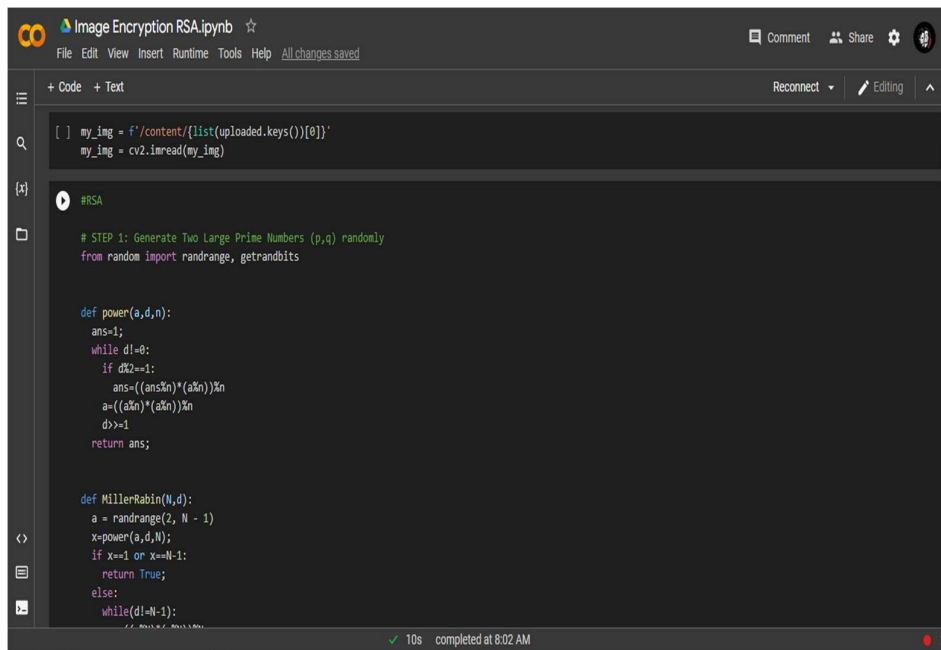
Choose files panda.jpg
• panda.jpg (image/jpeg) - 16085 bytes, last modified: 16/11/2022 - 100% done
Saving panda.jpg to panda.jpg
User uploaded file "panda.jpg" with length 16085 bytes

[ ] my_img = f'/content/{list(uploaded.keys())[0]}'
my_img = cv2.imread(my_img)

[ ] #RSA

10s completed at 8:02 AM
```

Fig 12: Implementing Code



```
Image Encryption RSA.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text Reconnect Editing

[ ] my_img = f'/content/{list(uploaded.keys())[0]}'
my_img = cv2.imread(my_img)

[ ] #RSA

# STEP 1: Generate Two Large Prime Numbers (p,q) randomly
from random import randrange, getrandbits

def power(a,d,n):
    ans=1;
    while d!=0:
        if d%2==1:
            ans=((ans*n)*(a%n))%n
            a=((a*n)*(a%n))%n
            d>>=1
        return ans;

def MillerRabin(N,d):
    a = randrange(2, N - 1)
    x=power(a,d,N);
    if x==1 or x==N-1:
        return True;
    else:
        while(d!=N-1):
            x=power(x,2,N);
            if x==1 or x==N-1:
                return True;
            d>>=1;
    return False;

10s completed at 8:02 AM
```

Fig 13: Implementing Code

```

def MillerRabin(N,d):
    a = randrange(2, N - 1)
    x=power(a,d,N);
    if x==1 or x==N-1:
        return True;
    else:
        while(d!=N-1):
            x=((x*x)*(x*x))%N;
            if x==1:
                return False;
            if x==N-1:
                return True;
            d<<=1;
        return False;

def is_prime(N,K):
    if N==3 or N==2:
        return True;
    if N<=1 or N%2==0:
        return False;

    #Find d such that d*(2^r)=N-1
    d=N-1
    while d%2!=0:
        d/=2;

```

10s completed at 8:02 AM

Fig 14: Implementing Code

```

[ ] #Find d such that d*(2^r)=N-1
d=N-1
while d%2!=0:
    d/=2;

for _ in range(K):
    if not MillerRabin(N,d):
        return False;
    return True;

def generate_prime_candidate(length):
    # generate random bits
    p = getrandbits(length)
    # apply a mask to set MSB and LSB to 1
    # Set MSB to 1 to make sure we have a Number of 1024 bits.
    # Set LSB to 1 to make sure we get a Odd Number.
    p |= (1 << length - 1) | 1
    return p

def generatePrimeNumber(length):
    A=4
    while not is_prime(A, 128):

```

10s completed at 8:02 AM

Fig 15: Implementing Code

```

def generatePrimeNumber(length):
    A=4
    while not is_prime(A, 128):
        A = generate_prime_candidate(length)
    return A

length=5
P=generatePrimeNumber(length)
Q=generatePrimeNumber(length)

print(P)
print(Q)

19
17

#Step 2: Calculate N=P*Q and Euler Totient Function = (P-1)*(Q-1)
N=P*Q
eulerTotient=(P-1)*(Q-1)
print(N)
print(eulerTotient)

323
288

```

10s completed at 8:02 AM

Fig 16: Implementing Code

```

#Step 3: Find E such that GCD(E,eulerTotient)=1(i.e., e should be co-prime) such that it satisfies this condition:- 1<E<eulerTotient

def GCD(a,b):
    if a==0:
        return b;
    return GCD(b%a,a)

E=generatePrimeNumber(4)
while GCD(E,eulerTotient)!=1:
    E=generatePrimeNumber(4)
print(E)

13

# Step 4: Find D.
#For Finding D: It must satisfies this property:- (D*E)%mod(eulerTotient)=1;
#Now we have two Choices
# 1. That we randomly choose D and check which condition is satisfying above condition.
# 2. For Finding D we can Use Extended Euclidean Algorithm: ax+by=1 i.e., eulerTotient(x)+E(y)=GCD(eulerTotient,e)
#Here, Best approach is to go for option 2.( Extended Euclidean Algorithm.)

def gcdExtended(E,eulerTotient):
    a1,a2,b1,b2,d1,d2=1,0,0,1,eulerTotient,E

    while d2!=1:

```

10s completed at 8:02 AM

Fig 17: Implementing Code

```

def gcdExtended(E, eulerTotient):
    a1, a2, b1, b2, d1, d2=1,0,0,1, eulerTotient, E

    while d2!=1:

        # k
        k=(d1//d2)

        #a
        temp=a2
        a2=a1-(a2*k)
        a1=temp

        #b
        temp=b2
        b2=b1-(b2*k)
        b1=temp

        #d
        temp=d2
        d2=d1-(d2*k)
        d1=temp

    D=b2

    if D>eulerTotient:
        D=D%eulerTotient
    elif D<0:

```

Fig 18: Implementing Code

```

D=b2

if D>eulerTotient:
    D=D%eulerTotient
elif D<0:
    D=D%eulerTotient

return D

D=gcdExtended(E, eulerTotient)
print(D)

133

[ ] row,col=my_img.shape[0],my_img.shape[1]
enc = [[0 for x in range(col)] for y in range(row)]

[ ] #Step 5: Encryption

for i in range(0,row):
    for j in range(0,col):
        r,g,b=my_img[i,j]
        C1=power(r,d,E,N)
        C2=power(g,d,E,N)
        C3=power(b,d,E,N)

```

Fig 19: Implementing Code

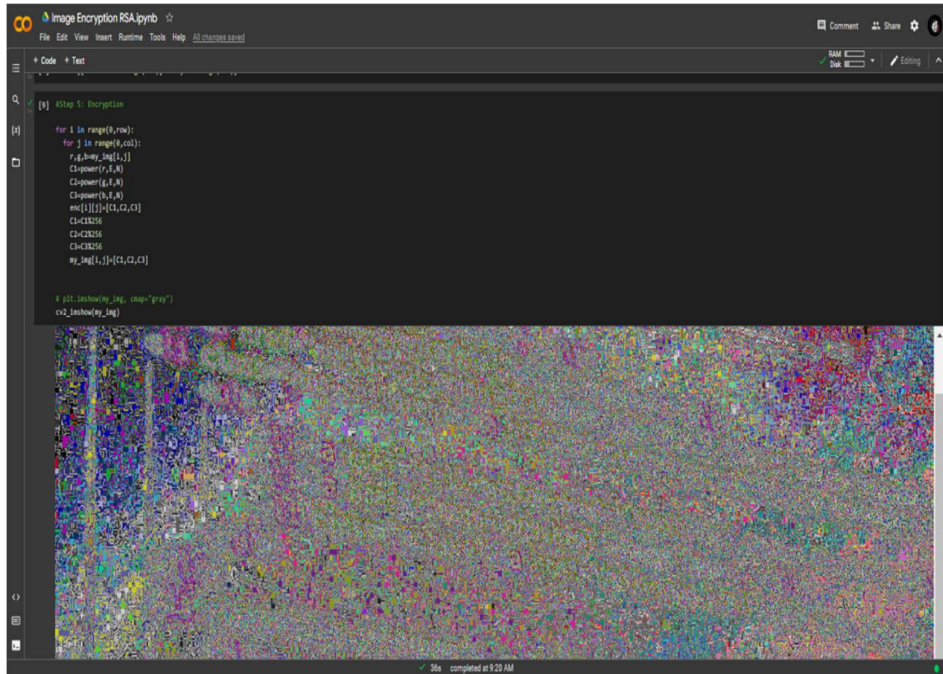


Fig 20: Implementing Code

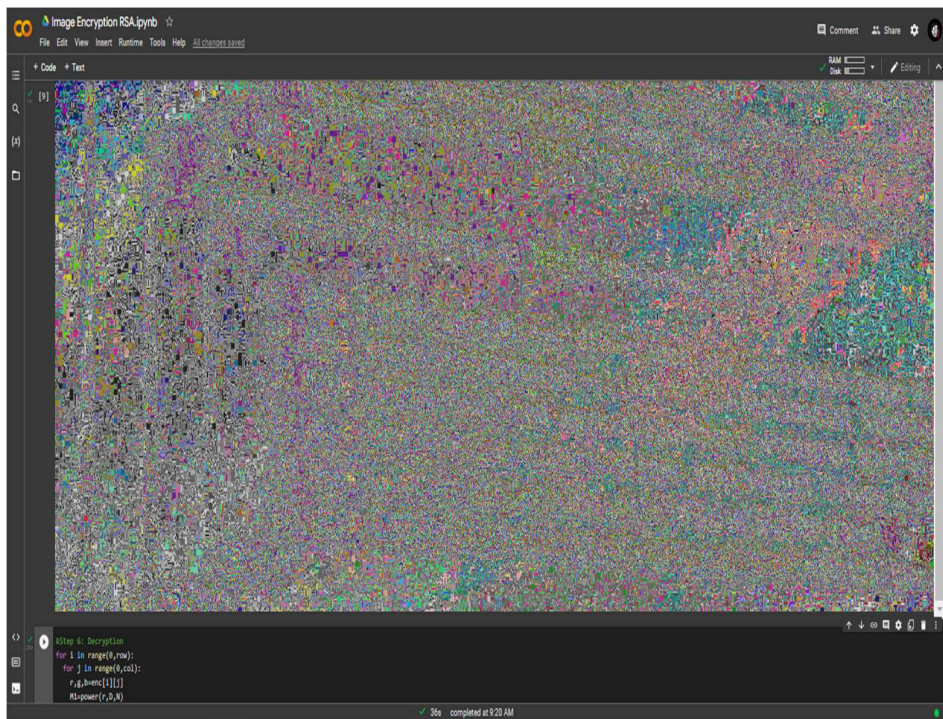


Fig 21: Encrypted Image

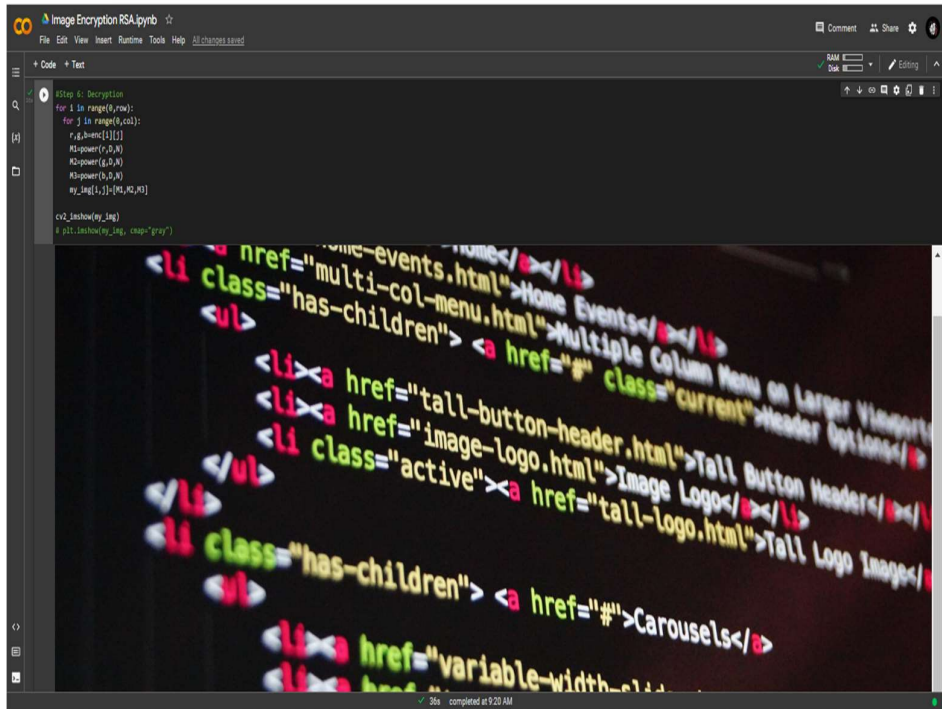


Fig 22: Decrypted Image

Outputs with different inputs

Table 2: P, Q, E, D, Time


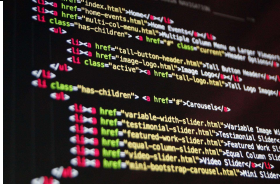
P	239	
Q	173	
E	27	
D	30323	
Time	3.774115	

Table 3: P, Q, E, D, Time


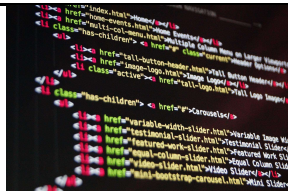
P	113	
Q	71	
E	6469	
D	589	
Time	4.240226	

Table 4: P, Q, E, D, Time

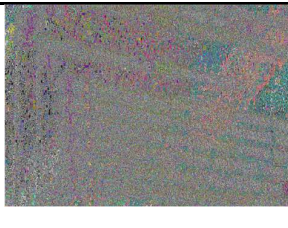
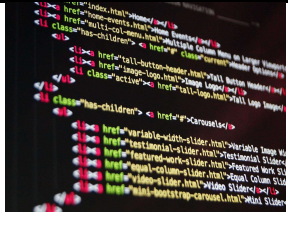
P	47	
Q	59	
E	21	
D	23885	
Time	3.762528	

Table 5: P, Q, E, D, Time

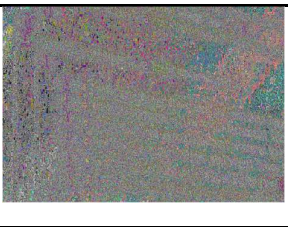
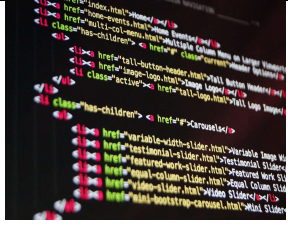

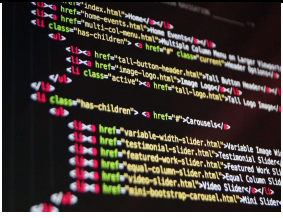
P	17	
Q	23	
E	109	
D	2661	
Time	3.681188	

Table 6: P, Q, E, D, Time

P	11	
Q	13	
E	47	
D	743	
Time	3.691191	

Chapter 05: CONCLUSIONS

5.1 Conclusions

RSA is the most commonly used encryption algorithm today, but there are certain limitations that must be considered in order for RSA to remain the best, and research needs to be done to make RSA anti-quantization. As the current the encryption system will soon be replaced, research in the area of quantum computer-resistant quantum encryption methods is needed more than ever. Developing qCrypt isn't enough, but it's a starting point. However, more research is needed on post-quantum cryptosystems.

5.2 Recommendations

A few recommendations should be emphasized regarding RSA encryption. RSA's strength lies in its large prime-based keys, but RSA is slow to generate keys compared to other algorithms.

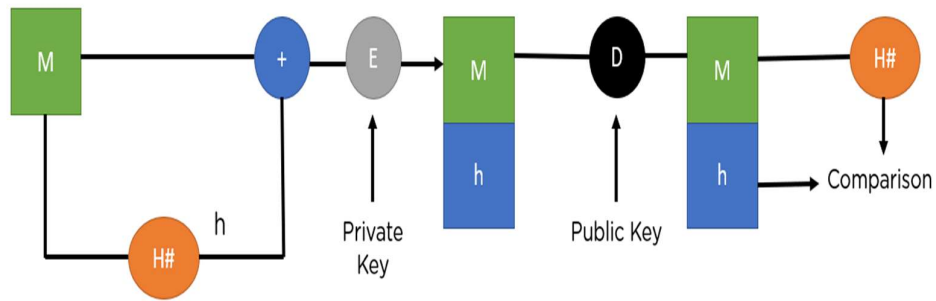
5.3 Applications

RSA encryption is generally used in combination with other encryption schemes, or for digital signatures which can validate the authenticity and integrity of a message. It cannot be used to encrypt entire messages or files, because it is less effective and more resource-heavy than symmetric-key encryption.

What are Digital Signatures?

Digital signatures serve the purpose of authentication and verification of documents and files. This is crucial to prevent tampering during official papers' transmission and prevent digital manipulation or forgery.

They work on the public key cryptography architecture, barring one small caveat. Typically, the asymmetric key system uses a public key for encryption and a private key for decryption. However, when dealing with digital signatures, it's the opposite. The private key is used to encrypt the signature, and the public key is used to decrypt it. Since the keys work in tandem with each other, decrypting it with the public key signifies it used the correct private key to sign the document, hence authenticating the origin of the signature.



M - Plain-text

H - Hash function

h - Hash digest

‘+’ - Bundle both plain text and digest

E - Encryption

D - Decryption

The image above shows the entire process, from the signing of the key to its verification. So, go through each step to understand the procedure thoroughly.

- Step 1: M denotes the original message It is first passed into a hash function denoted by H# to scramble the data before transmission.
- Step 2: It then bundles the message together with the hash digest, denoted by h, and encrypts it using the sender’s private key.
- Step 3: It sends the encrypted bundle of the message and digest to the receiver, who decrypt it using the sender’s public key.

- Step 4: Once decrypted, it passes the message through the same hash function (H#) to generate the hash digest again.
- Step 5: It compares the newly generated hash with the hash received in the decrypted bundle. If they match, it verifies the data integrity.

There are two industry-standard ways to implement the above methodology.

They are:

1. RSA Algorithm
2. DSA Algorithm

Both have the same goal, but they approach encryption and decryption in different ways.

REFERENCES

- <http://en.wikipedia.org/wiki/Encryption>
- <http://en.wikipedia.org/wiki/Plaintext>
- <http://en.wikipedia.org/wiki/Ciphertext>
- <http://en.wikipedia.org/wiki/Decryption>
- <http://en.wikipedia.org/wiki/Cryptography>
- F. H. M. S. Al-Kadei, H. A. Mardan and N. A. Minas, "Speed Up Image Encryption by Using RSA Algorithm," 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), 2020, pp. 1302-1307, doi: 10.1109/ICACCS48705.2020.9074430.
- Kaur, M., Kumar, V. A Comprehensive Review on Image Encryption Techniques. *Arch Computat Methods Eng* 27, 15–43 (2020). <https://doi.org/10.1007/s11831-018-9298-8>.
- Mai Helmy, El-Sayed M. El-Rabaie, Ibrahim Eldokany, F.E. Abd El-Samie, Proposed Hybrid Encryption Algorithm for Robust 3D Image Communication over Wireless Channels, *Optik*, 2022, 170205, ISSN 0030-4026, <https://doi.org/10.1016/j.ijleo.2022.170205>.
- Sahoo, A., Mohanty, P., Sethi, P.C. (2022). Image Encryption Using RSA Algorithm. In: Udgate, S.K., Sethi, S., Gao, XZ. (eds) *Intelligent Systems. Lecture Notes in Networks and Systems*, vol 431. Springer, Singapore. https://doi.org/10.1007/978-981-19-0901-6_56.
- https://www.researchgate.net/publication/307545291_Digital_Image_Encryption_Based_on_RSA_Algorithm.

APPENDICES

Code:

```
import cv2

import numpy as np

from google.colab.patches import cv2_imshow

import matplotlib.pyplot as plt

%matplotlib inline

from google.colab import files
uploaded = files.upload()

for fn in uploaded.keys():

    print ('User uploaded file "{name}" with length {length} bytes'.format(

        name=fn, length=len(uploaded[fn])))

my_img = f'/content/{list(uploaded.keys())[0]}'

my_img = cv2.imread(my_img)

#RSA
```


STEP 1: Generate Two Large Prime Numbers (p,q) randomly

```
from random import randrange, getrandbits
```

```
def power(a,d,n):
```

```
    ans=1;
```

```
    while d!=0:
```

```
        if d%2==1:
```

```
            ans=((ans%n)*(a%n)) %n
```

```
            a=((a%n)*(a%n)) %n
```

```
            d>>=1
```

```
    return ans;
```

```
def MillerRabin(N,d):
```

```
    a = randrange(2, N - 1)
```

```
    x=power(a,d,N);
```

```
    if x==1 or x==N-1:
```

```
        return True;
```

```
    else:
```

```

while(d! =N-1):

    x=((x%N)*(x%N)) %N;

    if x==1:

        return False;

    if x==N-1:

        return True;

    d<<=1;

return False;

def is_prime(N, K):

    if N==3 or N==2:

        return True;

    if N<=1 or N%2==0:

        return False;

    #Find d such that d*(2^r) =X-1

    d=N-1
    while d%2! =0:

```

```

    d/=2;

    for _ in range(K):

        if not MillerRabin(N,d):

            return False;

    return True;

def generate_prime_candidate(length):

    # Generate random bits

    p = getrandbits(length)

    # Apply a mask to set MSB and LSB to 1

    # Set MSB to 1 to make sure we have a Number of 1024 bits.

    # Set LSB to 1 to make sure we get an Odd Number.

    p |= (1 << length - 1) | 1

    return p

def generatePrimeNumber(length):

    A=4

    while not is_prime(A, 128):

```

```

    A = generate_prime_candidate(length)

    return A

length=5

P=generatePrimeNumber(length)

Q=generatePrimeNumber(length)

print(P)

print(Q)

#Step 2: Calculate  $N=P*Q$  and Euler Totient Function =  $(P-1) *(Q-1)$ 

N=P*Q

eulerTotient=(P-1) *(Q-1)

print(N)

print(eulerTotient)

#Step 3: Find E such that  $GCD(E,eulerTotient)=1$  (i.e., e should be co-prime)
such that it satisfies this condition: -  $1<E<eulerTotient$ 

def GCD(a,b):

```

```
if a==0:
```

```
    return b;
```

```
return GCD(b%a,a)
```

```
E=generatePrimeNumber(4)
```

```
while GCD(E,eulerTotient)!=1:
```

```
    E=generatePrimeNumber(4)
```

```
print(E)
```

Step 4: Find D.

```
#For Finding D: It must satisfy this property: - (D*E) Mod(eulerTotient)=1;
```

```
#Now we have two Choices
```

```
# 1. We randomly choose D and check which condition satisfies the above  
condition.
```

```
# 2. For Finding D we can Use Extended Euclidean Algorithm:  $ax+by=1$  i.e.,  
 $eulerTotient(x)+E(y)=GCD(eulerTotient,e)$ 
```

```
#Here, Best approach is to go for option 2. ( Extended Euclidean Algorithm.)
```

```
def gcdExtended(E,eulerTotient):
```

```
    a1, a2, b1, b2, d1, d2=1,0,0,1, eulerTotient, E
```

```
while d2!=1:
```

```
    # k
```

```
    k=(d1//d2)
```

```
    #a
```

```
    temp=a2
```

```
    a2=a1-(a2*k)
```

```
    a1=temp
```

```
    #b
```

```
    temp=b2
```

```
    b2=b1-(b2*k)
```

```
    b1=temp
```

```
    #d
```

```
    temp=d2
```

```
    d2=d1-(d2*k)
```

```
    d1=temp
```

```
    D=b2
```

```

if D>eulerTotient:

    D=D%eulerTotient

elif D<0:

    D=D+eulerTotient

return D

D=gcdExtended(E,eulerTotient)

print(D)

row,col=my_img.shape[0],my_img.shape[1]
enc = [[0 for x in range(col)] for y in range(row)]

#Step 5: Encryption
for i in range(0, row):

    for j in range(0, col):

        r,g,b=my_img[i,j]

        C1=power(r,E,N)

        C2=power(g,E,N)

        C3=power(b,E,N)

        enc[i][j]=[C1, C2, C3]

```

```
C1=C1%256
```

```
C2=C2%256
```

```
C3=C3%256
```

```
my_img[i,j]=[C1, C2, C3]
```

```
# plt.imshow(my_img, cmap="gray")
```

```
cv2.imshow(my_img)
```

#Step 6: Decryption

```
for i in range(0, row):
```

```
    for j in range(0, col):
```

```
        r,g,b=enc[i][j]
```

```
        M1=power(r,D,N)
```

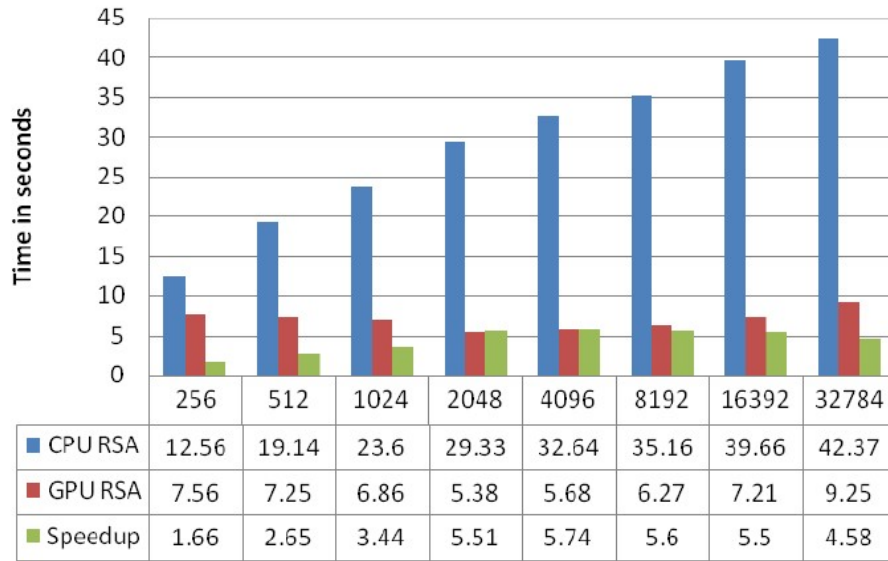
```
        M2=power(g,D,N)
```

```
        M3=power(b,D,N)
```

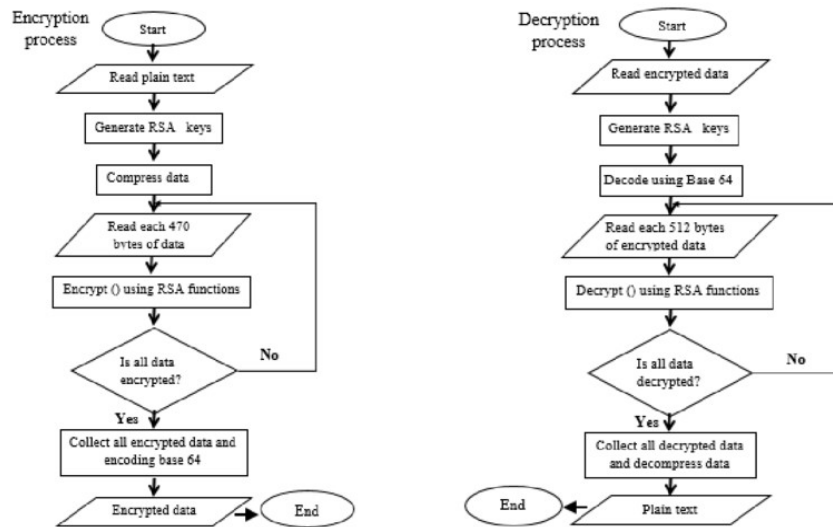
```
        my_img[i,j]=[M1, M2, M3]
```

```
cv2.imshow(my_img)
```

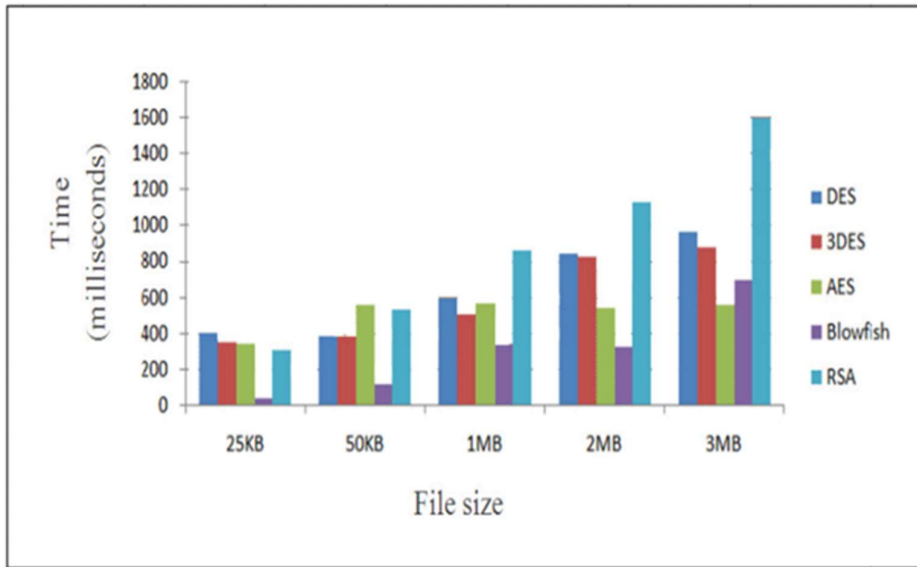
```
# plt.imshow(my_img, cmap="gray")
```

Graph 1: Graph showing effect of data input on CPU-RSA and GPU-RSA along with the Speedup



Graph 2: Flowchart-of-RSA-encryption-and-decryption-operations



Graph 3: Graph showing various encryption methods with time