Text and File Encryption and Decryption to Send Via Email using Double Layered Cryptosystem

Sarthi NIGAM, Vishrut TEWATIA, Ramesh SRIVASTAVA
Undergraduate, Delhi Technological University
Undergraduate, Delhi Technological University
Professor, Delhi Technological University

Abstract. With data and information being so easily accessible in the current digital age, we need to secure information stored in our computer or transmitted through the internet against various attacks. Cryptography is divided into the following two major types: Symmetric Key Cryptography and Asymmetric Key Cryptography. The height of security provided by the algorithm depends on the encryption and decryption keys. The primary information medium is E-Mail, which is highly susceptible to attacks and cryptography provides the backbone to its security. In this paper, we have worked on various algorithms such as Caesar, DES, AES, RSA and ECC for the encryption and decryption procedures and a new model on the basis of their study is devised, namely a Double Layered Cryptosystem which uses AES and RSA ciphers as its layers. For large files, the proposed system's encryption time becomes more comparable to the symmetric key ciphers while maintaining the security level of RSA cipher, being 92.65% faster than RSA and 94.83% faster than ECC. The model can be used for smaller files as well with slightly slower encryption speeds.

Keywords. Cryptography, Encryption, Decryption, Mail, Double Layered Cryptosystem

1. INTRODUCTION

E-mail is known to everyone as the new-age form of communication. Email has always been the way businesses communicate digitally with their customers and providers. A significant number of emails are shared each day. Hence, we can see the necessity of securing email and email transfer.

By default, emails aren't encrypted in the flow of traversal from sender's network to the receiver's network. Cyber offenders can easily break into your data and your emails as well as attachments are not safe. But, if the emails are encrypted, only the recipient for whom the email was meant can decrypt and access its details. There are two methods for email encryption in practice:

- Email Encryption while it's traversing through the network (TLS)
- Point-to-Point email Encryption

Cryptography is a way of securing interaction and communication channels between two parties with the use of a specified way, i.e., codes.

A symmetric key cipher is an algorithm that uses a single key for both the processes of encryption and decryption. The plaintext message conversion to ciphertext and vice versa are achieved using the aforementioned key, also called a private/secret key.

Asymmetric encryption is the other type of cryptographic algorithm which differs in the number and use of keys in its implementation from symmetric key cipher. Here, two keys, each is contained by the sender and the receiver and are called public and private key. The public key is available to all while the private key is solely available to the sender/receiver.

Our research expands on the use-case of these encryption algorithms and we devise an algorithm or a system to optimize speed as well as security in the transfer of files via mail through a mix of symmetric key and asymmetric key algorithms. Through comparison and analysis of various symmetric and asymmetric key ciphers such as Caesar, DES, AES, RSA and ECC, the performance of various algorithms is comprehensively studied. It helped us to introduce a new model, namely Double Layered Cryptosystem for email communication. The pre-defined methods had slight drawbacks like high encryption time and compromising the security for faster time but this model achieves to overcome these underlying setbacks. The model also gives the options for low as well as high security for preferences related to encryption speed. We have also implemented the inbuilt simple cipher which can prevent the original plaintext from getting revealed. So even simple mail exchanges could be normally encrypted with this simple cipher.

2. Literature review

2.1 Caesar Cipher

Category. Symmetric Key Cipher

Recognized as the shift cipher, Caesar Cipher is easily comprehensible and globally known. The technique used is substitution, meaning the plain text is substituted by some letter which is a fixed position away in the alphabet.

The Caesar cipher encryption function, $E(x)$, where x is the character to be encrypted, is represented as:

$$E(x) \equiv (x + k)\, mod\, n \tag{1}$$

where k is the key (basically the shift) used on each letter and n is the length of the character array (shown in Fig.1).

Similarly, the decryption function[1] is:

$$D(c) \equiv (c - k)\, mod\, n \tag{2}$$

where c is the cipher text or the encrypted character.

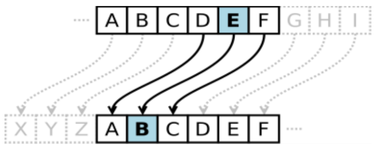


Figure 1. Caesar Cipher

2.2 Data Encryption Standard (DES)

Category. Symmetric Key Cipher

Rabah et al [2] gives a comprehensive review on the DES algorithm. It performs encryption on 64-bit chunks of plaintext to transform them into 64-bit chunks of ciphertext.

The Algorithm of Encryption of plaintext by DES (as seen in Fig.2):

- Initial Permutation: A specified permuting algorithm is used on the plaintext.
- The Key Transformation: The key size is 64-bits (plaintext size). It is compressed by ignoring the 8 parity bits present in it, which is every 8th bit. Hence, it is reduced to a size of 56-bits.
  Next, a different subkey is produced for all of the involved rounds (16) of the DES algorithm.
  Firstly, it is broken into two halves of 28-bits each. Then the left shift takes place in which one bit shift happens for round nos. 1,2,9 and 16 and the for the rest of the rounds, a left shift of two bits takes place.
  After this shifting, 48-bits of the 56-bits are chosen. The output is a 48-bit subkey (every round).
  - The Expansion Permutation: After producing the subkey, the plaintext is permuted. Now, the plaintext is divided into two equal 32-bit blocks.
    The main part of this step expands the right half of the plaintext to 48-bits. Finally, as the right half is now of the same size as the subkey, it is XORed with the subkey.
  - S-Box Substitution: The 48-bit resultant key from the last step then goes through the substitution step.
    This is performed by eight substitution boxes [3] or S-boxes. A 6-bit input is converted into a 4-bit output by the S-box. Therefore, the 48-bits are submerged into precisely eight sub blocks.
    Every S-box is a table of 4 ×16 dimensions in which each cell entry is a 4-bit number.
    LSB and MSB of the 6-bit input gives us a 2-bit number which when converted to decimal gives the position of the row. The 4-bits left in between the LSB and MSB when converted to decimal gives the position of the column and when tallied with the S-box, provides a resultant a 4-bit number as output.
    Combining all the S-boxes gives the 32-bit output.
  - P-Box Substitution: The 32-bit output of the right half is simply permuted and is then XORed with the 32-bit left half. Finally, the positions of the right and left halves are swapped and the next round starts from step 2(d).
  - Final Permutation: This is the reverse of the preliminary permutation which was done on the 64-bit plaintext.

The Decryption algorithm of DES occurs by utilising the keys obtained in the inverse order. So, Encryption keys $E_1, E_2 \ldots, E_{16}$ means decryption keys would be $E_{16}, E_{15}, \ldots, E_1$. Also, the left shift is reversed.

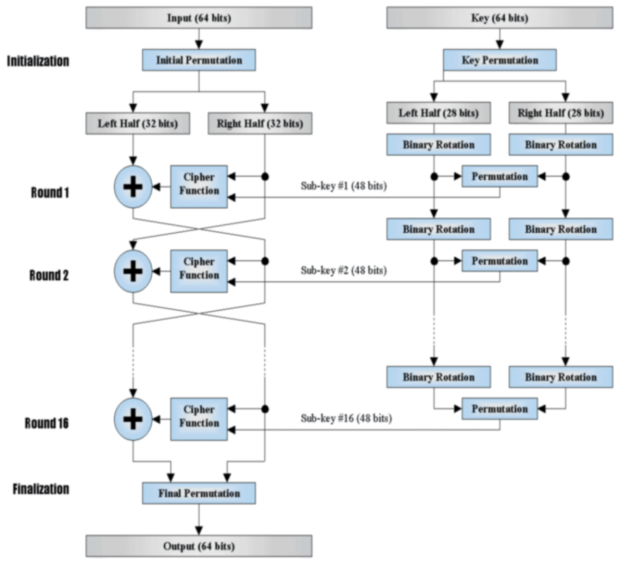


Figure 2. DES Algorithm

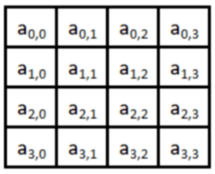
2.3 Advanced Encryption Standard (AES)

Category. Symmetric Key Cipher

Abdullah et al [4] discusses AES in depth. AES is more popular and widely known. Encryption of a 128-bit chunk of plaintext occurs in rounds of 10,12 and 14 which depends on the key size respectively (128-bits, 192 bits and 256-bits).

AES algorithm (refer Fig. 5) considers the 128-bit blocks as 16 bytes or 4 words, each word containing 4 bytes or 32 bits respectively. This leads to the formation of an input matrix of size 4×4.

- Formation of the state matrix:
  Input 4×4 matrix is transformed into another 4×4 matrix:
  Firstly, each column of the initial matrix is considered a word of 4 bytes. Let these be $w1, w2, w3$ and $w4$. Now, each cell is transformed.
  The first cell is transformed as $b_{0,0}$ meaning it's the 0th byte of 0th word, hence 8 bytes and so on (Fig. 3).

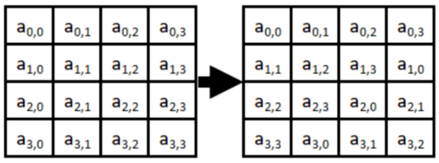


$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

Figure 3. State Matrix

Each round in the algorithm involves four sub-processes:

- Byte Substitution: S-box substitution takes place and the state matrix is transformed such that each cell of 1 byte or 8 bits is divided into two equal halves, the first 4-bits representing the row and next 4-bits representing the columns. The block in the S-box table is checked invoking the substitution to take place which results in a modified state matrix.
- Shifting rows: Now, each row of the state matrix undergoes left shift in a circular manner. 1st row isn't shifted, second row is left shifted by one place and so on (process shown in Fig. 4).



$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,0}$
$a_{2,2}$	$a_{2,3}$	$a_{2,0}$	$a_{2,1}$
$a_{3,3}$	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$

Figure 4. Shift Rows

- Mix Columns: Pre-defined input matrix is multiplied with each word of the row modified state matrix which gives a new word. Combining these words gives the new state matrix.
- Add Round-key: The input matrix is taken as 16 bytes and are XORed to the 16 bytes of the round key. The output 16 bytes are taken as input matrix for the next round. Finally, the ciphertext is obtained after all the rounds take place.

The Decryption process is inverse of the Encryption process [5]:

- Inverse shift rows
- Inverse sub bytes
- Add round key and then
- Inverse mix cols.

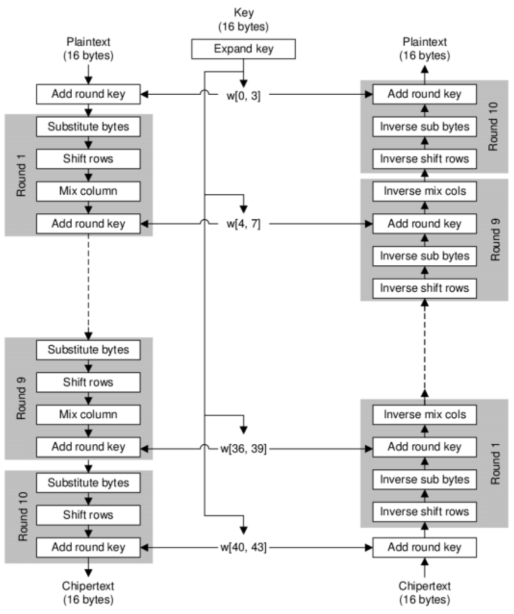


Figure 5. AES Algorithm

2.4 RSA Encryption

Category. Asymmetric Key Cipher

SARISAKAL et al [6] describes how to effectively develop RSA algorithm in JAVA. RSA (Rivest-Shamir-Adleman) is one the oldest and universally used algorithm for secure data transfer. RSA is fundamentally more complex and harder to implement, so in functionality, its much slower than the symmetric ones. Hence, only small quantities of data can be interpreted by it.

Taking two prime numbers into account, which are kept secret, a user can create a public key which can be used by anyone to encrypt the message but can only be decrypted by someone with knowledge of the prime numbers.

- Key-pair generation: Generation of the two keys, private and public keys.
  Two prime numbers p and q are chosen at random. p and q are kept secret from the public.
  $n = p \times q$ is calculated.
  n is considered as the modulus for the process of formation of both public and private keys. The length of it is actually the key length. It’s provided to everyone when the public key is released.
  $\Phi(n)$ is calculated, where Φ is the Euler's totient function.
  Euler's totient function basically returns back the number of co-prime numbers less than the input number in the function.
  Now $n = pq$,
  $\Phi(n) = \Phi(p) \times \Phi(q)$
  Since, p and q are prime, $\Phi(p) = p - 1$ and similarly $\Phi(q) = q - 1$.

Hence $\Phi(n) = (p-1) \times (q-1)$. $\Phi(n)$ is also reserved secret.
An integer e is selected satisfying $1 < e < \Phi(n)$ and $gcd(e, \Phi(n)) = 1$.
e is also provided to everyone when the public key is released.
d is obtained using $d = e^{-1}\, mod\, \Phi(n)$
d is reserved secret and is a component of the private key.
Public Key: Public elements- the modulus n and the encryption exponent e
Private Key: Private elements- decryption exponent d
p, q, and $\Phi(n)$ are also reserved secret since d can be obtained using them.

- Key exchange: A secret key is communicated securely which is used for encryption communication. Suppose that X wants to transfer information to Y. If they use the RSA algorithm for this transfer, X must know Y's public key to encrypt the message and to decrypt the message, Y should use their private key. For X to be able to share his/her encrypted messages, Y should share his/her public key (n, e) to X via a secure but open to the public, route.
- Encryption: Once X gets Y's public key, he/she can share a message M to Y. Now, M is turned into a suitable plaintext, m (which is an integer) which is padded. Then, the ciphertext c is calculated using Y's public key e, which is equivalent to

$$c \equiv m^e\, mod\, n \tag{3}$$

- Decryption: Similarly, Y can obtain m from c by using his/her private key exponent d using the following

$$m \equiv c^d\, mod\, n \tag{4}$$

Given m, Y can actually obtain the original message M by inversing the padding scheme used in step (3).

2.5 Elliptic Curve Cryptography (ECC)

Category. Asymmetric Key Cipher

Singh et al [7] discusses the method for text encryption using Elliptical curve cryptography (ECC). It's better than RSA encryption in the aspect that comparatively, the key size it requires is smaller and provides the same level of security.

Elliptic curve-based equations exhibit a characteristic that they are easy to calculate one-directionally hence generating a trapdoor situation.

An ECC (E) over a prime field p is defined by the given general equation in two variables with coefficients.

$$E: y^2 \equiv x^3 + ax + b\, mod\, p$$

$$where, 4a^3 + 27b^2 \neq 0 \tag{5}$$

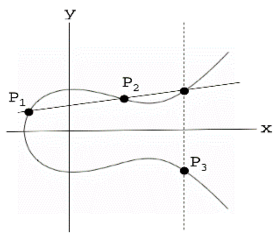


Figure 6. Elliptic Curve

The above condition ensures that there are no double roots present in the Elliptic curve E. If P_1 and P_2 are points on elliptic curve E (Fig. 6), then we can define

$$P_3 = P_1 + P_2 \tag{6}$$

Let us assume a point P on the curve. Recurrent addition takes place and the following operation is availed: $kP = P + P + P + \ldots + k$ times.

Let X & Y be the two interacting groups. A common EC equation is used and G, the generator point is incorporated.

- Key Pair Generation: Let X and Y private keys be nA and nB respectively. Both the selected private keys should be less than the modulus prime number p. X and Y public keys are:

$$P_X = nAG \text{ and } P_Y = nBG \tag{7}$$

- Encryption: Let the plaintext be defined as P_m. If X wants to transfer a message to Y, X has to encrypt the message using Y's public key (let P_Y). The ciphertext P_c is obtained by:

$$P_c = \{kG, P_m + kPY\} \tag{8}$$

where, k is an integer.

- Decryption: When Y wants to decrypt the ciphertext P_c, the following relational conversion takes place

$$P_m = \{P_c - nBkG\}$$

$$= \{P_m + kP - nBkG\}$$

$$= \{P_m + knBG - nBkG\} = P_m \tag{9}$$

Since the multiplier nB is the secret key P_Y of Y, only Y can decipher the plaintext transmitted by X.

3. Proposed Methodology

3.1 Performance of various algorithms to be chosen for modelling

JAVA programming language is used to simulate the algorithms. Files of diverse sizes are used to perform the simulations and execution, and comparison of four cryptography algorithms DES, AES, RSA and ECC is performed.

Table 1 shows the encryption time, considered the time for the complete conversion of a plaintext to ciphertext.

Table 1. Encryption Time of Algorithms

File Size	DES	AES	RSA	ECC
≤ 1 KB	5ms	3ms	41ms	48ms
100 KB	15ms	18ms	474ms	586ms
1 MB	161ms	159ms	1154ms	1772ms
10 MB	1104ms	892ms	13,657ms	19,413ms

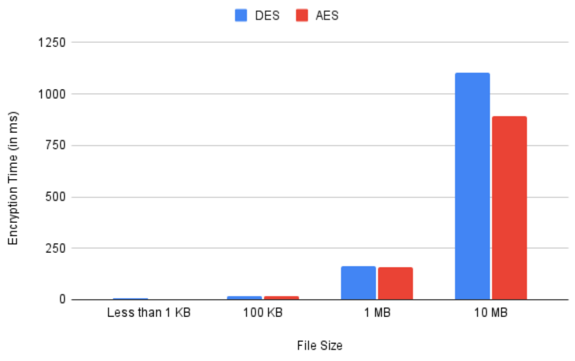


Figure 7. Analysis of Symmetric Key Ciphers

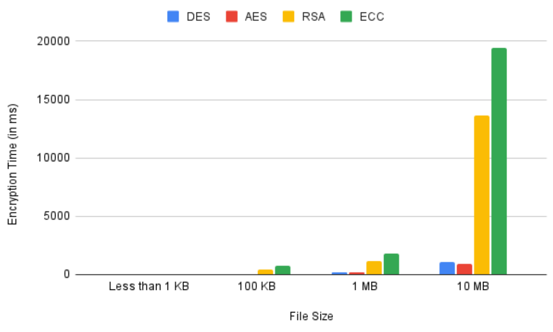


Figure 8. Analysis of Algorithms

3.2 Results obtained

From table 1, we can infer that the encryption time for ≤1 KB file has a considerably smaller difference than the difference in encryption time for large files. Also, the encryption speed of Symmetric Key Ciphers is comparably greater than the Asymmetric Key Ciphers for the large size of files (Fig. 7, Fig. 8). While symmetric key ciphers are faster, security of asymmetric key ciphers is greater than the symmetric key ciphers. Since RSA is as fast as AES for very small file size and more secure than AES, we can implement a Double Layered Cryptosystem which can be more secure and comparably fast as symmetric key ciphers.

4. MODEL

In a Double Layered Cryptosystem along with the inbuilt simple cipher which can prevent the original plaintext from getting revealed.

We will use the AES algorithm to encrypt and decrypt the plaintext or files whereas the RSA algorithm will be used to encrypt and decrypt the key of the AES algorithm which will be comparably smaller than the size of the file.

The overall process of the cryptosystem we devised will be as follows:

- Key Generation: Generate the keys required for the AES and RSA algorithm as previously mentioned and store the RSA keys for later use.
- Encryption: The Encryption process will be done in two layers.
- Base Layer: The base layer would consist of encryption of the plaintext or the given file by the above-mentioned AES encryption.
- Surface Layer: In this layer, the encryption will be performed on the key of the AES algorithm, i.e., the plaintext for the RSA algorithm is the key of the AES algorithm.
- Decryption: Similarly, the decryption will be done in two layers but in the reverse order. First, the AES algorithm key will be decrypted with the help of RSA private key[8] and finally the ciphertext will then be decrypted by the AES key obtained by the decryption through RSA algorithm.

Thus, this double layered cryptosystem will use the properties of both symmetric and asymmetric key ciphers. The model will be developed by using the combination of RSA algorithm which is more secure and AES algorithm which is faster. So, this proposed model is expected to give encryption time comparable to the AES and security to the level of the RSA algorithm.

4.1 Pseudo PlainText

Sometimes, the keys can be disclosed due to some mis happenings and the cyber-attackers now have both: the ciphertext and the key. To prevent the original text falling into their hands, we can implement and inbuilt Caesar cipher or some XOR function that the attackers cannot know until they look into the source code of the project.

5. SIMULATION PROCESS

After comparison and analysis of different cryptography algorithms, we observe that encryption time is similar for both symmetric and asymmetric key ciphers. Therefore, we have designed a Graphical User Interface (GUI) that any user can use to encrypt or decrypt a file that has to be sent via email.

5.1 Creating a GUI and simulating the process

For the model, we will be implementing the cryptosystems on JAVA and the hardware used are 2.8 GHz Octa-Core Intel Core i7 processor and 8 GB 2400 MHz Dual-Channel DDR4 RAM. Our main objective is to create an environment where the sender and recipient can encrypt and decrypt the information they have. For the simulation, we will be using files in a size range from 32 B to 10 MB.

Therefore, our main course of action will be as follows:

- Implementation of various algorithms, both symmetric and asymmetric key ciphers
- Comparison of encryption times of the algorithms on a given file.
- Analyze the change in encryption time of different types and sizes of files.
- Aim to develop an optimized algorithm for better security and fast encryption.
- Create a GUI which a User can use to encrypt or decrypt a given file.

5.2 Text encryption and decryption

For the text encryption and decryption, since the input data is of String format, the overall size of the data is small. So, we have implemented the RSA algorithm for this reason.

- Input Text: It takes text as an input; it can be either plaintext or ciphertext based on what one has to perform: Encryption or Decryption.
- Encrypt Button: It generates the keys of the algorithm and stores both the key and ciphertext file in a directory for the encryption process.
- Decrypt Button: After clicking, a window pops up which allows you to choose the key and the cipher file that has to be decrypted.
- Output Text: It displays the result of the process performed on button click. In case of decryption, if the input text and the ciphertext file are not same then it displays error.

5.3 File encryption and decryption

For the text encryption and decryption, we have two choices- low security and high security.

- Low Security: We have implemented the AES algorithm for this choice (it has low security).
- High Security: We have implemented double layered cryptosystem as it requires high security as well as faster encryption as files can be of large size.

The overall functions of the File Encryption-Decryption panel (Fig. 9) are as follows:

- Input File Location Button: This button allows us to choose file that has to be encrypted/decrypted and displays the file path in the below textbox.

- High Security Button: On clicking, it implements the double layered cryptosystem.
- Low Security Button: On clicking, it implements the AES algorithm.
- Encrypt Button: This button has to be clicked to perform the encryption. It then generates the keys of the algorithm and stores both the key and file in a directory.
- Decrypt Button: This button is used for performing the decryption process. After clicking, a window pops up which allows you to choose the key and the file that has to be decrypted.
- Output File Location: It displays the path file of the resultant file created on button click.

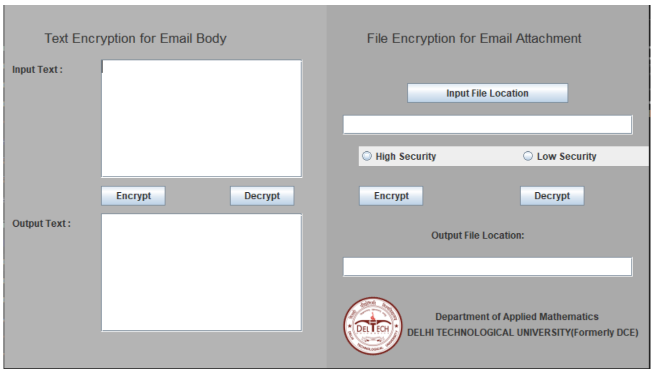

Text Encryption for Email Body
Input Text :
Encrypt
Decrypt
Output Text :
File Encryption for Email Attachment
Input File Location
High Security
Low Security
Encrypt
Decrypt
Output File Location:
Department of Applied Mathematics
DELHI TECHNOLOGICAL UNIVERSITY(Formerly DCE)

Figure 9. User Interface

6. Results

The Table 2 shows the comparison of the encryption time of the double layered cryptosystem with respect to the four algorithms on different sizes of files.

Table 2. Encryption Time of Algorithms

File Size	DES	AES	RSA	ECC	Double Layered
≤ 1 KB	5ms	3ms	41ms	48ms	54ms
100 KB	15ms	18ms	474ms	586ms	127ms
1 MB	161ms	159ms	1154ms	1772ms	308ms
10 MB	1104ms	892ms	13,657ms	19,413ms	1003ms

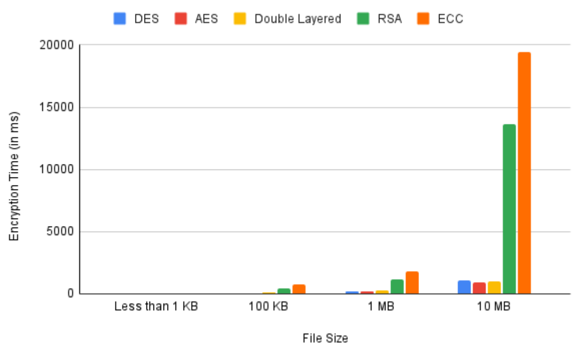


Figure 10. Comparison with other algorithms

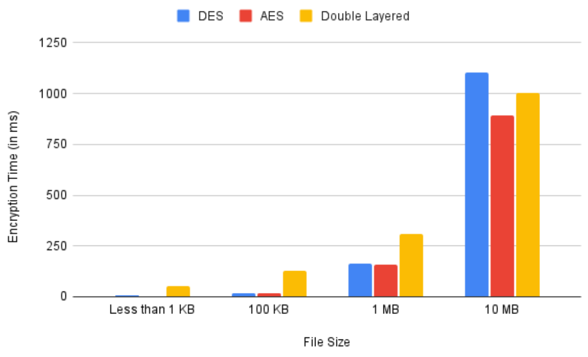


Figure 11. Comparison with Symmetric Key Ciphers

- For <= 1 KB files, encryption time is comparable to asymmetric key ciphers.
- For 100 KB files, the proposed system is 73.20% faster than RSA and 78.32% faster than ECC.
- For 1 MB files, the proposed system is 73.31% faster than RSA and 82.62% faster than ECC.
- For 10MB files, the proposed system is 9.15% faster than DES and 12.44% slower than AES as compared to being 92.65% faster than RSA and 94.83% faster than ECC.

7. CONCLUSION

Through comparison and analysis, the performance of various algorithms- DES, AES, RSA and ECC is comprehensively studied. It helped us to introduce a new model, namely Double Layered Cryptosystem for email communication, implemented in JAVA language on VS Studio Platform. We have also implemented the inbuilt simple cipher which can prevent the original plaintext from getting revealed. The experimental results show that as the file size increases, the proposed system's encryption time becomes more comparable to the symmetric key ciphers while maintaining the security level of RSA cipher, being 9.15% faster than DES and just 12.44% slower than AES in the case of 10MB files as compared to being 92.65% faster than RSA and 94.83% faster than ECC. Comparing 100KB and 10 MB file results, the system's encryption time became faster by 21% and 17.41% for RSA and ECC ciphers respectively. Hence, for large files, this system will provide encryption time comparable to symmetric key ciphers and security as high as asymmetric key ciphers. Further, the system could be studied to see the security level as compared to popular mail encryption techniques like SSL or Secure Sockets Layer (used by Yahoo) and others existing in the current IT security ecosystem.

REFERENCES

[1] Oppliger, R. (2005). *Contemporary cryptography*. Artech House.

[2] Rabah, Kefa. "Theory and implementation of data encryption standard: A review." Information Technology Journal 4, no. 4 (2005): 307-325.

[3] Baignères Thomas, & Vaudenay, S. (2011). Conventional Cryptography. In *A classical introduction to cryptography*. essay, Springer.

[4] Abdullah, AkoMuhamad. "Advanced encryption standard (aes) algorithm to encrypt and decrypt data." Cryptography and Network Security 16 (2017).

[5] M Gajbhiye, V., & V G, P. (2015). Implementation of rijindael's encryption and decryption algorithm using FPGA. *International Journal of Electronics and Communication Engineering*, *2*(5), 67–70. https://doi.org/10.14445/23488549/ijece-v2i5p121

[6] SARISAKAL, M. Nusret, Selcuk SEVGEN, and A. C. A. R. Dogal. "Developing An Application of RSA Algorithm With JAVA." In ELECO'2001 International Conference on Electrical and Electronics Engineering, pp. 7-11. 2001.

[7] Singh, Laiphrakpam Dolendro, and Khumanthem Manglem Singh. "Implementation of text encryption using elliptic curve cryptography." Procedia Computer Science 54 (2015): 73-82.

[8] S. Jianjun, L. Ming and M. Jingang, "Research and application of data sharing platform integrating Ethereum and IPFs Technology," 2020 19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES), 2020, pp. 279-282, doi: 10.1109/DCABES50732.2020.00079.