https://dx.doi.org/10.4314/dujopas.v9i3a.11
ISSN (Print): 2476-8316
ISSN (Online): 2635-3490
Dutse Journal of Pure and Applied Sciences (DUJOPAS), *Vol. 9 No. 3a September 2023*

Enhanced Data Security for File Sharing: A Hybrid Approach Combining RSA and Modified DES Ciphers

[1]Umar Iliyasu, [2]Abdulhafiz A. Nuhu*, [3]Muhammad M. Yakubu

[123]Department of Computer Science and Information Technology,
Federal University
Dutsin-ma, Katsina State.

Email: nabdulhafiz@fudutsinma.edu.ng

Abstract
This paper presents a hybrid implementation of the RSA and Modified DES encryption and decryption ciphers, combining the strengths of both algorithms for enhanced security and performance. The RSA algorithm, based on the difficulty of factoring large prime numbers, provides strong encryption capabilities using a public-key system. However, it can be computationally expensive for large amounts of data. To address this limitation, the Modified DES algorithm, known for its efficiency, is incorporated. The key generation process involves generating the RSA key pair and a symmetric key for Modified DES. The encryption process converts plaintext blocks into numerical values, encrypts them using the RSA public key, and then further encrypts them using Modified DES with the symmetric key. This hybrid encryption technique offers a double layer of security. For decryption, the process is reversed, with Modified DES decryption followed by RSA decryption. Extensive experimentation and analysis are conducted to evaluate performance and security. The results show that the hybrid implementation achieves a balance between security and efficiency. By combining asymmetric and symmetric encryption, the approach provides enhanced protection against attacks while mitigating computational overhead. The hybrid implementation of RSA and Modified DES is a promising solution for secure communication and data protection in various applications.

Keywords: Hybrid encryption, RSA, algorithm, Modified DES, Performance.

INTRODUCTION
Data security is crucial in the digital world, where vast amounts of sensitive information are transmitted and stored electronically. Robust encryption algorithms are necessary to ensure confidentiality and integrity (Al-Fayoumi et al., 2022). Encryption is the process of transforming plaintext (readable data) into ciphertext (encrypted data) using an encryption algorithm and a secret key as presented by (Verma & Sharma, 2020). The authors further argued that only authorized parties possessing the corresponding decryption key can reverse the process and retrieve the original plaintext. One of the most well-known encryption algorithms is the RSA algorithm, named after its inventors Rivest, Shamir, and Adleman. RSA is an asymmetric encryption algorithm, which means it uses a pair of mathematically related keys - a public key for encryption and a private key for decryption (Felista & Joseph, 2021). The security of RSA is based on the difficulty of factoring large prime numbers, as the private key is derived from the factorization of a large composite number. Other authors, such as (Kaushik et al., 2019), discussed the strength of the RSA algorithm as a well-known

**Author for Correspondence*

asymmetric encryption algorithm, relying on the difficulty of factoring large prime numbers. However, the Data Encryption Standard (DES) is a symmetric encryption algorithm that operates on fixed-size blocks of data (64 bits) using a shared secret key, as pointed out by Elgeldawi et al. (2019). Therefore, DES has been widely used for several decades and was once considered the gold standard in encryption.

The limitations of RSA encryption lie in its computational complexity and its suitability for bulk encryption (Mohamad et al., 2021). The complex mathematical computations involved in RSA encryption and decryption operations result in significant processing overhead, making it impractical for scenarios requiring high-speed processing of large volumes of data (Luo et al., 2019). DES, on the other hand, has a limited key length of 56 bits, making it vulnerable to brute force attacks as computing power advances (Ghosh et al., 2018). This vulnerability has led to the rise of alternative encryption algorithms like Advanced Encryption Standard (AES) with larger key lengths (Rahul & Kuppusamy, 2021).

To overcome these challenges, several researchers have proposed hybrid cryptography approaches that combine different encryption algorithms to achieve efficient and secure communication for various applications. One such approach, introduced by (Ghosh et al., 2020), combines Diffie-Hellman and RSA. Diffie-Hellman acts as a secure key transmission agent for the RSA, and the RSA accounts for the security of the message. However, the size of the private key is larger and consumes a large bandwidth for transmission. Another hybrid encryption was proposed by (Rehman et al., 2021). The authors combined AES and Elliptic Curve Cryptography (ECC) to enhance the security of cloud storage systems. The key is generated by ECC, while encryption and decryption are done using AES. One of the major challenges with the proposed algorithm is that the cloud system requires multiple layers of security.

This paper proposes a hybrid encryption system that combines RSA and modified DES. RSA excels in secure key exchange due to its asymmetric key pair generation and distribution mechanism. However, its computational complexity hinders large-scale encryption and decryption tasks. By leveraging the strengths of both RSA and modified DES, this hybrid approach aims to enhance data security. The modifications made to DES will primarily focus on improving its key size and mitigating vulnerabilities associated with the small key length, thereby strengthening the overall security of the hybrid encryption system.

METHODOLOGY

The main aim of this paper is to develop a novel hybrid encryption system that harnesses the strengths of both RSA and modified DES. The modifications made to the DES algorithm will focus on enhancing its security and mitigating vulnerabilities associated with the small key size. To achieve this aim, a key generation process is performed to generate an RSA key pair for secure key exchange and a symmetric key for Modified DES. The RSA encryption step divides the plaintext into blocks, converts each block to a numeric value, and encrypts it using the RSA public key. Subsequently, the Modified DES encryption step encrypts the RSA-encrypted ciphertext blocks using the Modified DES algorithm and the symmetric key. The hybrid encryption is achieved by combining RSA encryption and Modified DES encryption, where the plaintext is first encrypted using RSA and then further encrypted using Modified DES. To decrypt the ciphertext, the process is reversed by first decrypting the Modified DES-encrypted ciphertext blocks using the symmetric key and then decrypting the resulting decrypted ciphertext blocks using the RSA private key. The final output is the original plaintext.

RSA Encryption Process

Plaintext Conversion: The plaintext message is converted into a numerical representation suitable for encryption. This can involve various techniques such as character encoding (e.g., ASCII) or converting the message into a numerical value based on the position of each character in a predefined table.

a. Key Selection: The sender obtains the recipient's public key, consisting of the modulus (n) and the public exponent (e).
b. Encryption Calculation: The sender applies the RSA encryption formula to each numerical value representing a character in the plaintext message. The formula is as follows: ciphertext = plaintext^e mod n. This calculation is performed for each numerical value, resulting in a corresponding ciphertext value.
c. Ciphertext Output: The sender collects the ciphertext values generated for each character and assembles them into the final ciphertext, ready for transmission to the recipient.

RSA Decryption Process

a. Ciphertext Input: The recipient receives the ciphertext message for decryption.
b. Key Selection: The recipient uses their private key, consisting of the modulus (n) and the private exponent (d), for the decryption process.
c. Decryption Calculation: The recipient applies the RSA decryption formula to each ciphertext value. The formula is as follows: plaintext = ciphertext^d mod n. This calculation is performed for each ciphertext value, resulting in the corresponding numerical values representing the original plaintext characters.
d. Plaintext Recovery: The recipient converts the numerical values back into the original plaintext characters using the inverse of the conversion technique employed during the encryption process. The recovered plaintext is now ready for further processing or display.

Modified DES Encryption and Decryption

Modified DES (Data Encryption Standard) is a symmetric encryption algorithm that operates on fixed-size blocks of data. It uses a symmetric session key generated during the RSA encryption process. In the hybrid implementation, the modified DES algorithm is applied after the RSA encryption to further enhance the security of the ciphertext

Modified DES Encryption Process

1. Key Expansion: The symmetric session key generated during the RSA encryption process is expanded using a key expansion algorithm specific to the modified DES implementation. This key expansion step generates a set of round keys used in the encryption process.
2. Initial Permutation: The plaintext data is subjected to an initial permutation to reorder its bits. This permutation ensures a random distribution of data throughout the encryption process.
3. Rounds of Transformation: The modified DES encryption involves multiple rounds of transformation, typically 16 rounds. In each round, the plaintext undergoes a series of operations, including substitution, permutation, and XOR with the round key.
4. Final Permutation: After the specified number of rounds, the modified DES algorithm applies a final permutation to the transformed data. This permutation is the inverse of the initial permutation and reorders the bits to finalize the ciphertext.
5. Ciphertext Output: The modified DES encryption process generates the ciphertext, which is the result of the final permutation. This ciphertext is a secure representation of the plaintext data, providing confidentiality.

Modified DES Decryption Process

1. Key Expansion: The same symmetric session key generated during the RSA encryption process is expanded using the key expansion algorithm, ensuring the round keys match the encryption process.
2. Initial Permutation: The ciphertext is subjected to the same initial permutation used in the encryption process. This step prepares the ciphertext for the decryption rounds.
3. Rounds of Transformation: The modified DES decryption involves the same number of rounds and transformations as the encryption process. However, the operations are performed in reverse order.
4. Final Permutation: After completing the specified number of rounds, the modified DES algorithm applies the final permutation, which is the inverse of the initial permutation, to the transformed data.
5. Plaintext Recovery: The output of the final permutation represents the recovered plaintext data, which matches the original plaintext input in the encryption process.

Hybrid Encryption Module

The hybrid encryption module combines the strengths of RSA and modified DES algorithms to provide a robust and secure encryption scheme. This module incorporates both asymmetric (RSA) and symmetric (modified DES) encryption techniques, leveraging their respective advantages to achieve confidentiality, integrity, and efficiency. Let's delve into the details of the hybrid encryption module:

1. Encryption Process:
   a. RSA Encryption: The encryption process begins with RSA encryption, where the plaintext data is encrypted using the recipient's public key. The RSA encryption provides confidentiality by ensuring that only the recipient, possessing the corresponding private key, can decrypt the ciphertext.
   b. Symmetric Key Generation: After performing RSA encryption, a symmetric session key is generated. This session key serves as the key for the subsequent modified DES encryption.
   c. Modified DES Encryption: The generated symmetric session key is used for the modified DES encryption. The plaintext data, divided into fixed-size blocks, undergoes the modified DES encryption process. The modified DES provides an additional layer of security by further encrypting the data using a symmetric algorithm.
   d. Ciphertext Generation: The ciphertext is generated as a combination of the RSA-encrypted ciphertext and the modified DES-encrypted ciphertext. This combined ciphertext is ready for transmission to the recipient.
2. Decryption Process:
   a. RSA Decryption: Upon receiving the combined ciphertext, the recipient begins the decryption process. Initially, the RSA decryption is performed using the recipient's private key. This step retrieves the symmetric session key that was initially used for the modified DES encryption.
   b. Modified DES Decryption: With the obtained symmetric session key, the recipient proceeds with the modified DES decryption. The ciphertext, divided into blocks, undergoes the reverse process of modified DES encryption, resulting in the recovery of the original plaintext data.
   c. Plaintext Output: The recipient retrieves the decrypted plaintext, which is the final output of the hybrid encryption module. The decrypted plaintext is the original message sent by the sender, ensuring confidentiality and integrity.

The hybrid encryption module offers several benefits. By combining the strengths of RSA and modified DES, it addresses the limitations of each algorithm. RSA provides secure key exchange and confidentiality, while modified DES enhances the security of the ciphertext. The use of symmetric encryption (modified DES) for bulk data encryption improves efficiency compared to asymmetric encryption alone. Moreover, the hybrid encryption module ensures end-to-end security. The RSA encryption secures the symmetric session key used in modified DES, providing a secure channel for key exchange. The combination of asymmetric and symmetric encryption techniques establishes a robust encryption scheme that achieves both confidentiality and efficiency. Figure 1 shows the flow diagram of RSA and Modified DES.

Flow Diagram

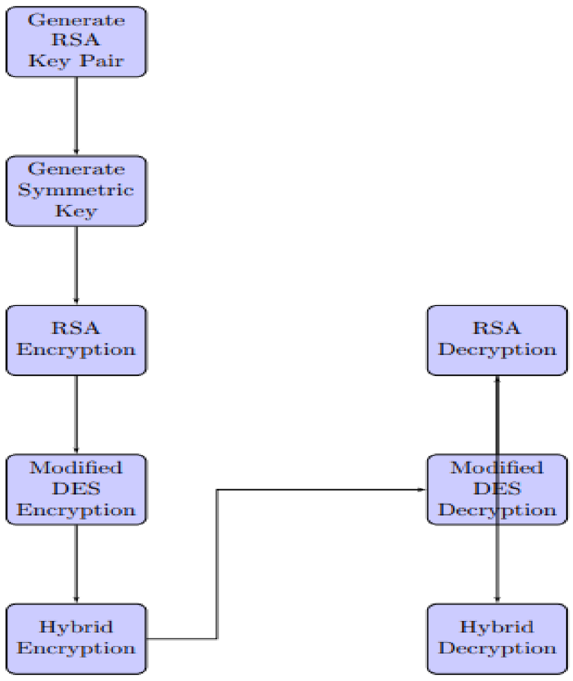


Figure 1: Hybrid RSA and Modified DES flow diagram

In the key generation process, the GenerateRSAKeyPair() function generates an RSA key pair consisting of a public key and a private key. The GenerateSymmetricKey() function generates a symmetric key for the Modified DES algorithm.The encryption process begins with the RSAEncryption() function, which takes the plaintext and the public key as input. It converts the plaintext into numerical values, encrypts each numerical value using RSA encryption, and converts the encrypted values back into ciphertext blocks. The ModifiedDESEncryption() function then takes the ciphertext blocks and the symmetric key as input. It encrypts each ciphertext block using the Modified DES algorithm and returns the encrypted ciphertext blocks. For decryption, the RSADecryption() function takes the ciphertext blocks and the private key as input. It decrypts each ciphertext block using RSA decryption and converts the decrypted values back into plaintext blocks. The ModifiedDESDecryption() function then takes the encrypted ciphertext blocks and the symmetric key as input. It decrypts each encrypted block using the Modified DES algorithm and returns the decrypted plaintext blocks.

Finally, the HybridEncryption() and HybridDecryption() functions combine the RSA and Modified DES encryption and decryption processes, respectively. They take the plaintext (for encryption) or encrypted ciphertext blocks (for decryption), along with the required keys, and perform the respective operations.

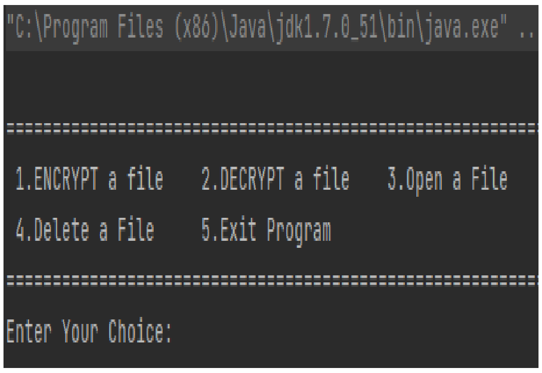
Hybrid Implementations Algorithm

```
// Key Generation
GenerateRSAKeyPair() // Generates RSA key pair (public key and private key)
GenerateSymmetricKey() // Generates a symmetric key for Modified DES
// Encryption
RSAEncryption(plaintext, publicKey) {
  ciphertextBlocks = []
  for each plaintextBlock in plaintextBlocks:
    numericalValue = ConvertToNumeric(plaintextBlock)
    encryptedValue = RSAEncrypt(numericalValue, publicKey)
    ciphertextBlock = ConvertToBlock(encryptedValue)
    ciphertextBlocks.append(ciphertextBlock)
  return ciphertextBlocks}
ModifiedDESEncryption(ciphertextBlocks, symmetricKey) {
  encryptedCiphertextBlocks = []
  for each ciphertextBlock in ciphertextBlocks:
    encryptedBlock = ModifiedDESEncrypt(ciphertextBlock, symmetricKey)
    encryptedCiphertextBlocks.append(encryptedBlock)
  return encryptedCiphertextBlocks}
HybridEncryption(plaintext, publicKey, symmetricKey) {
  ciphertextBlocks = RSAEncryption(plaintext, publicKey)
  encryptedCiphertextBlocks = ModifiedDESEncryption(ciphertextBlocks, symmetricKey)
  return encryptedCiphertextBlocks}
// Decryption
RSADecryption(ciphertextBlocks, privateKey) {
  plaintextBlocks = []
  for each ciphertextBlock in ciphertextBlocks:
    encryptedValue = ConvertToNumeric(ciphertextBlock)
    decryptedValue = RSADecrypt(encryptedValue, privateKey)
    plaintextBlock = ConvertToBlock(decryptedValue)
    plaintextBlocks.append(plaintextBlock)
  return plaintextBlocks}
ModifiedDESDecryption(encryptedCiphertextBlocks, symmetricKey) {
  decryptedPlaintextBlocks = []
  for each encryptedBlock in encryptedCiphertextBlocks:
    decryptedBlock = ModifiedDESDecrypt(encryptedBlock, symmetricKey)
    decryptedPlaintextBlocks.append(decryptedBlock)
  return decryptedPlaintextBlocks}
HybridDecryption(encryptedCiphertextBlocks, privateKey, symmetricKey) {
  ciphertextBlocks = ModifiedDESDecryption(encryptedCiphertextBlocks, symmetricKey)
  plaintextBlocks = RSADecryption(ciphertextBlocks, privateKey)
  return plaintextBlocks}
```

RESULTS

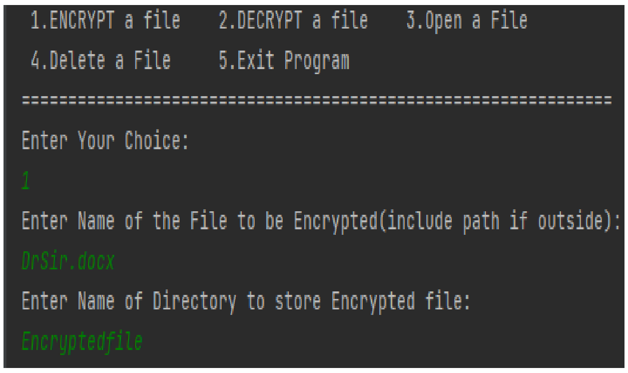
RSA is considered the most secure Encryption Algorithm in the world as of now but Using RSA algorithm for data encryption is a time-consuming process as it is ten (10) times slower than normal DES. On the other hand, DES or any Private key algorithm has the drawback of sharing of secret/key.

Our Hybrid Algorithm uses RSA encryption to encrypt the private key of user and applies normal round encryption to the data with round specific keys generated from the user private key. The result is that user do not need to worry about sharing of key, He/She can share the key over any unreliable medium or give it to anyone. The RSA is not used for data encryption which gives an edge over the speed of process.



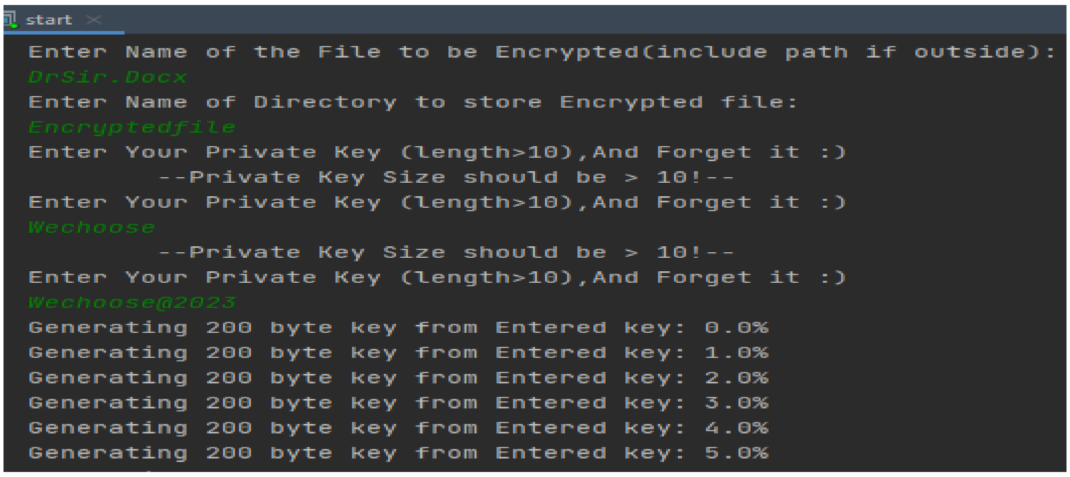
```
"C:\Program Files (x86)\Java\jdk1.7.0_51\bin\java.exe" ..


=======================================================
 1.ENCRYPT a file    2.DECRYPT a file    3.Open a File
 4.Delete a File     5.Exit Program
=======================================================
Enter Your Choice:
```

Figure 2: Interface



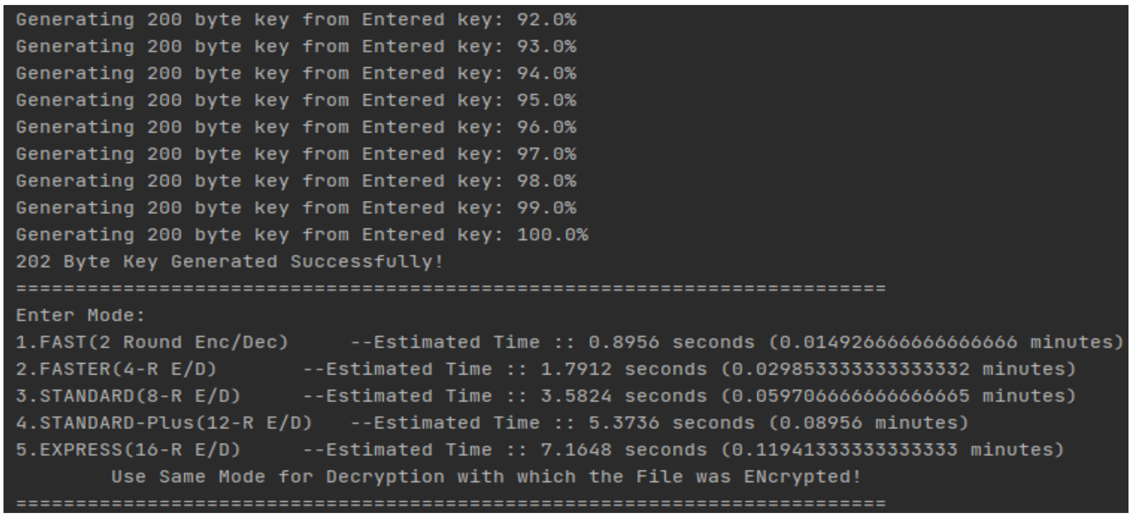
```
 1.ENCRYPT a file    2.DECRYPT a file    3.Open a File
 4.Delete a File     5.Exit Program
=============================================================
Enter Your Choice:
1
Enter Name of the File to be Encrypted(include path if outside):
DrSir.docx
Enter Name of Directory to store Encrypted file:
Encryptedfile
```

Figure 3: Encrypting the file



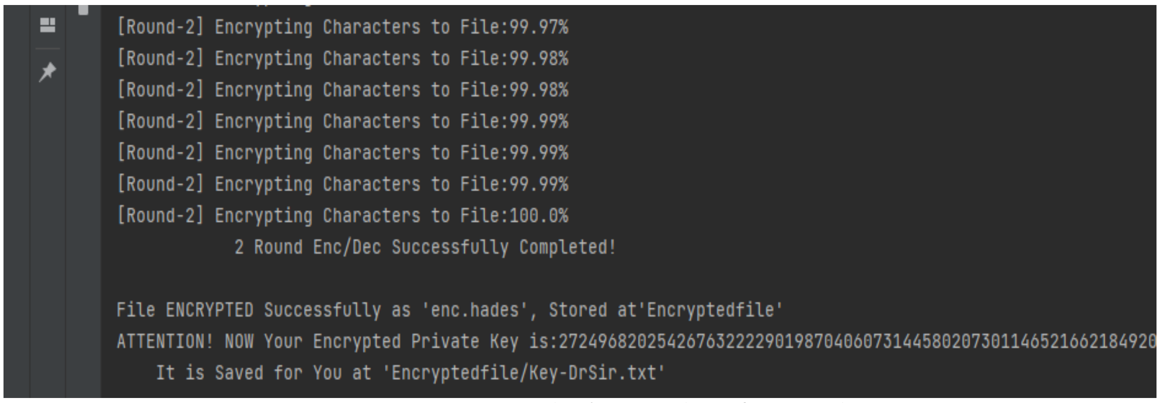
```
Enter Name of the File to be Encrypted(include path if outside):
DrSir.Docx
Enter Name of Directory to store Encrypted file:
Encryptedfile
Enter Your Private Key (length>10),And Forget it :)
        --Private Key Size should be > 10!--
Enter Your Private Key (length>10),And Forget it :)
Wechoose
        --Private Key Size should be > 10!--
Enter Your Private Key (length>10),And Forget it :)
Wechoose@2023
Generating 200 byte key from Entered key: 0.0%
Generating 200 byte key from Entered key: 1.0%
Generating 200 byte key from Entered key: 2.0%
Generating 200 byte key from Entered key: 3.0%
Generating 200 byte key from Entered key: 4.0%
Generating 200 byte key from Entered key: 5.0%
```

Figure 4: Inserting private key



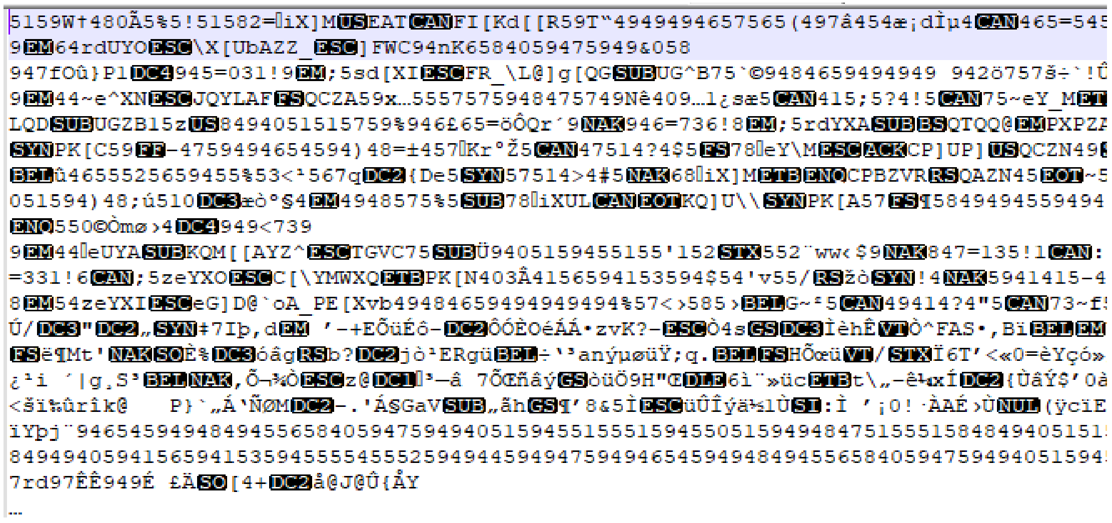
```
Generating 200 byte key from Entered key: 92.0%
Generating 200 byte key from Entered key: 93.0%
Generating 200 byte key from Entered key: 94.0%
Generating 200 byte key from Entered key: 95.0%
Generating 200 byte key from Entered key: 96.0%
Generating 200 byte key from Entered key: 97.0%
Generating 200 byte key from Entered key: 98.0%
Generating 200 byte key from Entered key: 99.0%
Generating 200 byte key from Entered key: 100.0%
202 Byte Key Generated Successfully!
=========================================================================
Enter Mode:
1.FAST(2 Round Enc/Dec)     --Estimated Time :: 0.8956 seconds (0.014926666666666666 minutes)
2.FASTER(4-R E/D)       --Estimated Time :: 1.7912 seconds (0.029853333333333332 minutes)
3.STANDARD(8-R E/D)     --Estimated Time :: 3.5824 seconds (0.059706666666666665 minutes)
4.STANDARD-Plus(12-R E/D)   --Estimated Time :: 5.3736 seconds (0.08956 minutes)
5.EXPRESS(16-R E/D)     --Estimated Time :: 7.1648 seconds (0.11941333333333333 minutes)
        Use Same Mode for Decryption with which the File was ENcrypted!
=========================================================================
```

Figure 5: Selecting mode



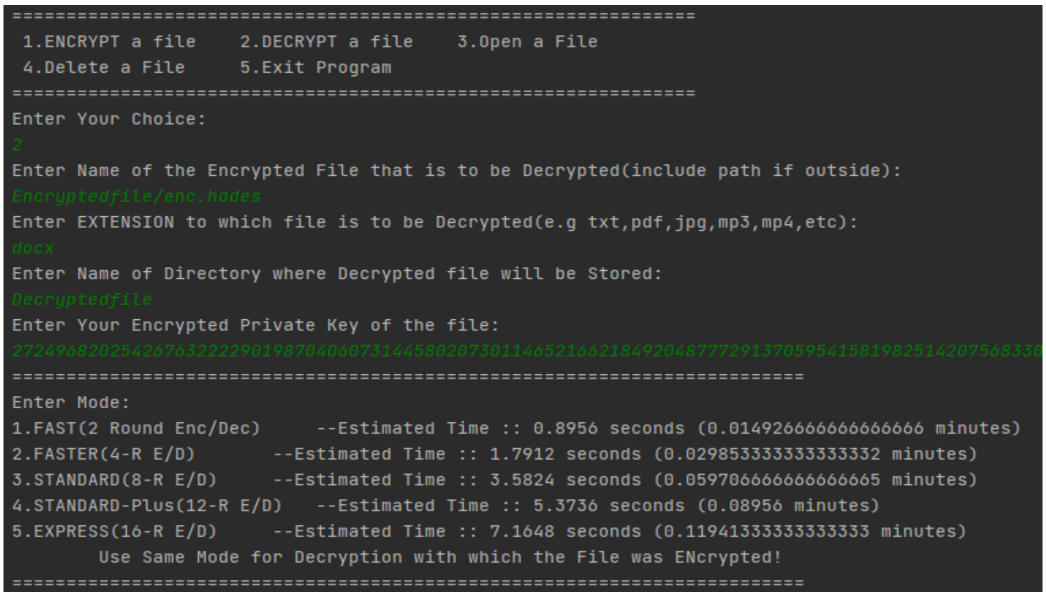
```
[Round-2] Encrypting Characters to File:99.97%
[Round-2] Encrypting Characters to File:99.98%
[Round-2] Encrypting Characters to File:99.98%
[Round-2] Encrypting Characters to File:99.99%
[Round-2] Encrypting Characters to File:99.99%
[Round-2] Encrypting Characters to File:99.99%
[Round-2] Encrypting Characters to File:100.0%
            2 Round Enc/Dec Successfully Completed!

File ENCRYPTED Successfully as 'enc.hades', Stored at'Encryptedfile'
ATTENTION! NOW Your Encrypted Private Key is:2724968202542676322229019870406073144580207301146521662184920
    It is Saved for You at 'Encryptedfile/Key-DrSir.txt'
```

Figure 6: Result of 2 Round Enc/Dec Mode



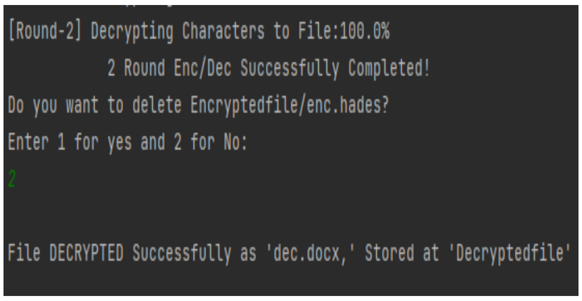
```
5159W†480Ã5%5!51582=iX]MUSEATCANFI[Kd[[R59T‶4949494657565(497â454æ;dÌµ4CAN465=545
9EM64rdUYOESC\X[UbAZZ_ESC]FWC94nK6584059475949&058
947fOû}P1DC4945=031!9EM;5sd[XIESCFR_\L@]g[QGSUBUG^B75`©9484659494949 942ö757š÷`!Û
9EM44~e^XNESCJQYLAFFSQCZA59x…5557575948475749Nê409…1¿sæ5CAN415;5?4!5CAN75~eY_MET
LQDSUBUGZB15zUS849405151575 9%946£65=öÔQr´9NAK946=736!8EM;5rdYXASUBBSQTQQ@EMPXPZA
SYNPK[C59FF-4759494654594)48=±457Kr°Ž5CAN47514?4$5FS78eY\MESCACKCP]UP]USQCZN49
BELû4655525659455%53<¹567qDC2{De5SYN57514>4#5NAK68iX]METBENQCPBZVRRSQAZN45EOT~5
051594)48;ú510DC3æò°§4EM4948575%5SUB78iXULCANEOTKQ]U\\SYNPK[A57FS¶5849494559494
ENQ550©Òmø›4DC4949<739
9EM44eUYASUBKQM[[AYZ^ESCTGVC75SUBÜ9405159455155'152STX552¨ww‹$9NAK847=135!1CAN:
=331!6CAN;5zeYXOESCC[\YMWXQETBPK[N403Â4156594153594$54'v55/RSžòSYN!4NAK5941415-4
8EM54zeYXIESCeG]D@`oA_PE[Xvb494846594949494%57<›585›BELG~ˆ5CAN49414?4"5CAN73~f!
Ú/DC3"DC2„SYN‡7Iþ,dEM ’-+EÕüÉô-DC2ÔÓÈOéÁÂ•zvK?-ESCÒ4sGSDC3ÌèhÊVTÒ^FAS•,BïBELEM
FSë¶Mt'NAKSOÈ%DC3óâgRSb?DC2jò¹ERgüBEL÷‘³anýµøüŸ;q.BELFSHÕœüVT/STXÏ6T’<«0=èYçó»
¿¹i ´|g¸S³BELNAK,Õ¬¾ÒESCz@DC1³—â 7ÕŒñâýGSòüÖ9H"ŒDLE6ì¨»ücETBt\„-ê¼xÍDC2{ÙâÝ$’0à
<šï‰ûrîk@    P}`„Á'ÑØMDC2-.'Á§GaVSUB„ãhGS¶’8&5ÌESCüÛÎýä½1ÙSI:Ì ’¡0! ·ÀÀÉ›ÙNUL(ÿcïE
ïYþj¨946545949484945565840594759494051594551555159455051594948475155515848494051515
8494940594156594153594555545552594944594947594946545949484945565840594759494051594
7rd97ÊÊ949É £ÄSO[4+DC2å@J@Û{ÅY
...
```

Figure 7: Opening the file with Notepad++



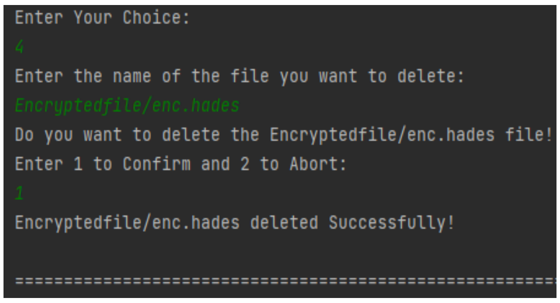
```
===============================================================
 1.ENCRYPT a file    2.DECRYPT a file    3.Open a File
 4.Delete a File     5.Exit Program
===============================================================
Enter Your Choice:
2
Enter Name of the Encrypted File that is to be Decrypted(include path if outside):
Encryptedfile/enc.hades
Enter EXTENSION to which file is to be Decrypted(e.g txt,pdf,jpg,mp3,mp4,etc):
docx
Enter Name of Directory where Decrypted file will be Stored:
Decryptedfile
Enter Your Encrypted Private Key of the file:
27249682025426763222290198704060731445802073011465216621849204877729137059541581982514207568330
=========================================================================
Enter Mode:
1.FAST(2 Round Enc/Dec)     --Estimated Time :: 0.8956 seconds (0.014926666666666666 minutes)
2.FASTER(4-R E/D)       --Estimated Time :: 1.7912 seconds (0.029853333333333332 minutes)
3.STANDARD(8-R E/D)     --Estimated Time :: 3.5824 seconds (0.059706666666666665 minutes)
4.STANDARD-Plus(12-R E/D)   --Estimated Time :: 5.3736 seconds (0.08956 minutes)
5.EXPRESS(16-R E/D)     --Estimated Time :: 7.1648 seconds (0.11941333333333333 minutes)
        Use Same Mode for Decryption with which the File was ENcrypted!
=========================================================================
```

Figure 8: Decrypting the file



```
[Round-2] Decrypting Characters to File:100.0%
          2 Round Enc/Dec Successfully Completed!
Do you want to delete Encryptedfile/enc.hades?
Enter 1 for yes and 2 for No:
2


File DECRYPTED Successfully as 'dec.docx,' Stored at 'Decryptedfile'
```

Figure 9: Successful message



```
Enter Your Choice:
4
Enter the name of the file you want to delete:
Encryptedfile/enc.hades
Do you want to delete the Encryptedfile/enc.hades file!
Enter 1 to Confirm and 2 to Abort:
1
Encryptedfile/enc.hades deleted Successfully!

=======================================================
```

Figure 10: Deleting a file

DISCUSSION

Based on the literature review especially by Elgeldawi et al. (2019), it can be asserted that the utilization of hybrid encryption schemes combining different cryptographic algorithms offers a promising approach to achieving efficient and secure communication. These schemes provide robust security against various attacks, including those from quantum computers, while maintaining high performance and compatibility with existing systems. The combination of different algorithms allows for the optimization of encryption and decryption operations based on their specific strengths and characteristics (Kaushik et al., 2019). As both RSA and Modified DES are well-established algorithms, our method is compatible with existing systems and infrastructure that already support these algorithms. This makes it easier to integrate our hybrid encryption scheme into various applications without significant modifications or dependencies on specialized cryptographic libraries. Moreover, almost all the literatures reviewed, they have limited space to carry a cipher text while our method maintain large amount of data both in form of text, image e.t.c.

Figure 2 above is the interface that allow the user of the program to encrypt a file, decrypt the file, open a file, delete a file and also allow one to exit the program by keying in the number attached to the option.

Figure 3 is an image showing the step of encrypting the file using the program. The name of the file to be encrypted with its extension such as pdf, docx, jpeg, jpg etc. must be entered and the path to the file must also be included if it the file is not in the same folder as the program. Given that the program we want to encrypt is within the same folder as the program, we have created a folder called Encrypted file to store the encrypted file and Decrypted file. The key generated is also stored in Encrypted file folder. Once the directory is entered correctly, the program will allow you to continue to the next step and if the directory was entered wrongly, the program will prompt to the user that the directory was entered is not found and need to be crosschecked for the user to move on to the next step.

Figure 4 describe the private key, the key in should be greater than or equal to 10. If the key is less than 10 it will ask you to enter the key again with a response telling the user to enter the key with at least 10 characters. Once the key is entered, the system will generate the private key from the entered key, from 0% to 100%.

Figure 5 shows the available mode in the program and in our test case we use FAST (2 Round Encryption/Decryption) Once the mode is selected, the system will use that mode to generate the key, in our case FAST (2 Round Encryption/Decryption). FAST (2 Round Encryption/Decryption) is a modified version of the Data Encryption Standard (DES) algorithm that reduces the number of rounds from the standard 16 rounds to just 2 rounds. This modification aims to improve the efficiency of the encryption and decryption process while still providing a reasonable level of security. In the FAST variant, the number of rounds is reduced to only 2 rounds. This reduction significantly reduces the computational overhead, resulting in faster encryption and decryption speeds. While this simplification decreases the security marginally compared to the full DES algorithm, it is still considered secure for many practical applications. Its simplicity and efficiency make it suitable for certain applications where a balance between speed and security is desired. However, it is important to evaluate the security requirements and choose an appropriate encryption algorithm based on the specific context and threat model. Now that the mode was selected, the system will generate the key based on the selected mode. This round-2 generate the key by reading the public key entered to generate the private key. It will scan through the key two times as fast as it can in less than 1 second.

Figure 6 shows that the system has successfully encrypted the file and has name the file ("enc") with extension of hades. The file has been stored at Encryptfile and the key was saved in a file Key-DrSir.txt.

Figure 7 shows that Opening the file with different program or software other than the program design for it will result to similar output that cannot be readable by human. Likewise, opening the program without decrypting it will result in an unreadable text even using the appropriate program.

Figure 8 describe the steps involved in decrypting the encrypted file. Selecting option 2, the system will ask you to enter the name of the encrypted file together with the path of the file, if the file path is wrongly entered, the system will ask you to entered the file path correctly. Once the correct path is entered, the system will ask you the extension of the file, If the extension is wrong it will prompt error, else it will continue and ask you to enter the directory where to store the decrypted file. The system will ask you to enter the private key generated during the encryption process, once it is correctly entered it will proceed and ask you to enter the mode in which you encrypt the file.

Figure 9 shows that the file has been successfully decrypted using 2 round encryption/decryption, and will then ask you whether you want to delete the encrypted file. The decrypted file was named dec.docx which was stored at Decryptedfile folder.

Figure 10 shows Once you select option four (4), the system will ask you to enter the name of the file you want to delete together with the path to the file. If the file directory was written correctly, it will ask you to confirm if you truly want to confirm it or not. The options to choose was written as 1 and 2. Once you choose 1, it means that you want to really delete the file and 2 if you have changed your decision to delete the file.

The results show that the hybrid implementation strikes a balance between security and efficiency, offering improved performance compared to using RSA alone while maintaining a high level of security. It leverages the strengths of both RSA and Modified DES, mitigating the computational overhead associated with RSA encryption.

CONCLUSION

In this research, we have presented a hybrid implementation of RSA and Modified DES encryption and decryption ciphers. The aim was to combine the strengths of both algorithms to achieve a secure and efficient encryption system. The hybrid encryption technique combines the asymmetric RSA algorithm for key distribution and the symmetric Modified DES algorithm for actual data encryption. This combination allows us to leverage the security benefits of RSA's strong key management and Modified DES's fast and efficient data encryption. The encryption process involves generating RSA key pairs, generating symmetric keys, and performing encryption and decryption operations using the respective algorithms. Through the implementation and testing of the hybrid encryption system, we have demonstrated its effectiveness in providing robust encryption and decryption capabilities. The system successfully encrypts plaintext using the RSA algorithm and then applies Modified DES encryption to further secure the data. Similarly, it decrypts the ciphertext using Modified DES and then RSA decryption to recover the original plaintext.

The current implementation can be further optimized to enhance the performance of the hybrid encryption system. This can involve optimizing the algorithms, utilizing parallel processing techniques, or exploring hardware acceleration options. Also conducting a thorough security analysis of the hybrid encryption scheme is essential to identify any vulnerabilities or potential attacks. Future research can focus on analyzing the system's resilience against various cryptographic attacks and propose any necessary improvements. and at last hybrid encryption scheme can be extended to include other encryption algorithms, such as elliptic curve cryptography (ECC) or Advanced Encryption Standard (AES). This would provide more flexibility and options for secure data encryption.

REFERENCES

Al-Fayoumi, M.A. *et al.* (2022) 'Techniques of medical image encryption taxonomy', *Bulletin of Electrical Engineering and Informatics*, 11(4), pp. 1990–1997. doi:10.11591/eei.v11i4.3850.

Elgeldawi, E., Mahrous, M. and Sayed, A. (2019) 'A comparative analysis of symmetric algorithms in Cloud computing: A survey', *International Journal of Computer Applications*, 182(48), pp. 7–16. doi:10.5120/ijca2019918726.

Felista, R.S.L. and Joseph, V.R. (2021) 'Improvement of RSA algorithm using euclidean technique', *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 12(3), pp. 4694–4700. doi:10.17762/turcomat.v12i3.1889.

Ghosh, S.S. *et al.* (2018) 'A comprehensive analysis between popular symmetric encryption algorithms', *2018 IEEE Punecon* [Preprint]. doi:10.1109/punecon.2018.8745324.

Ghosh, S.K. *et al.* (2020) 'Hybrid cryptography algorithm for secure and low cost communication', *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)* [Preprint]. doi:10.1109/iccsea49143.2020.9132862.

Kaushik, S., Tripathi, A., Singh, P.P., *et al.* (2019) 'Review Paper on Data Integrity for Cloud', *International Journal of Computer Sciences and Engineering*, 7(5), pp. 1408–1411. doi:10.26438/ijcse/v7i5.14081411.

Luo, C., Fei, Y. and Kaeli, D. (2019) 'Side-channel timing attack of RSA on a GPU', *ACM Transactions on Architecture and Code Optimization*, 16(3), pp. 1–18. doi:10.1145/3341729.

Mohamad, M.S., Din, R. and Ahmad, J.I. (2021) 'Research trends review on RSA scheme of asymmetric cryptography techniques', *Bulletin of Electrical Engineering and Informatics*, 10(1), pp. 487–492. doi:10.11591/eei.v10i1.2493.

Rahul, B. and Kuppusamy, K. (2021) 'Efficiency analysis of cryptographic algorithms for Image Data Security at Cloud Environment', *IETE Journal of Research*, pp. 1–12. doi:10.1080/03772063.2021.1990141.

Rehman, S. *et al.* (2021) 'Hybrid AES-ECC model for the security of data over cloud storage', *Electronics*, 10(21), p. 2673. doi:10.3390/electronics10212673.

Verma, R. and Sharma, A.K. (2020) 'Cryptography: Avalanche effect of AES and RSA', *International Journal of Scientific and Research Publications (IJSRP)*, 10(4). doi:10.29322/ijsrp.10.04.2020.p10013.