Advanced-Data Encryption using three-layered Hybrid Cryptosystem and Secured key storing using Steganography

Tanmoy Sen Gupta
B.Tech. Student, Department of Computer Science and Engineering
SRM Institute of Science and Technology

Ritik Srivastava
B.Tech. Student, Department of Computer Science and Engineering
SRM Institute of Science and Technology

Abstract

Security of Data is one of the most important aspects of one's digital presence. With the advent of new and sophisticated technologies, existing data security systems are becoming less efficient in protecting data. Thus the requirement of a multi-layered Hybrid Cryptosystem is ever more necessary. In this paper, a hybrid data encryption protocol is implemented and studied for its efficiency and performance. The hybrid crypto-system implements a combination of Blowfish, RSA, and AES layers to encrypt data. The system also encrypts the keys used and embeds them in an image using LSB Steganography. This research studies a potential hybrid-cryptosystem that may be able to address the drawbacks of existing traditional systems.

Keywords: *Encryption, Decryption, Hybrid Cryptography, AES, RSA, Blowfish*.

Introduction

Various Encryption Algorithms are used in apps and services to secure data. But the advent of new and sophisticated technologies is making these existing systems obsolete. Advancements in Hardware have significantly reduced the time required to break a cryptographic system. The proposed system utilizes a combination of three of the most robust and popular algorithms to secure data. A combination of Asymmetric Cryptography Algorithm RSA and Symmetric Cryptography Algorithms AES and Blowfish. RSA is one of the most widely used asymmetric encryption algorithms, that is it requires two separate keys to encrypt and decrypt, over the net, specifically on the TLS Layer [1], and used for various other functions apart from encryption of data. Blowfish and AES, on the other hand, are Symmetric Ciphers, that is, it uses only one key for both encryption and decryption. While Blowfish is the Fastest Encryption algorithm [2], AES is the most secure and efficient in encrypting data [3]. A combination of these can help in addressing the drawbacks of their standalone counterparts.

Studies showed that a Hybrid Cryptosystem of AES-Blowfish had higher effectiveness of encryption as compared to traditional standalone counterparts [4]. A similar cryptosystem consisting of RSA and AES combination implemented in Java was studied and was found to provide a high level of security as well as enhanced integrity and the use of LSB Steganography to store data in image in the system showed that the system was resistant to attacks as the histograms of both the cover and stego image were similar [5].

The proposed system in this paper uses a layered encryption architecture that encrypts data thrice using the three different algorithms and to ensure key security, the keys used are also encrypted and stored in an image using steganography. The keys are encrypted using the hash of the password, as the key for AES. SHA-1 is used to generate the hash from the user input password. The proposed system implemented in Python has proven to be a viable cryptosystem for securing data based on experimental results.

Algorithms Used

Blowfish Algorithm

Blowfish is a Symmetric Block Cipher designed in 1993 by B. Schneier. Blowfish algorithm has a block size of 64 Bits and key lengths varying from 32 to 448 Bits. It is particularly known for its features like complicated key schedules and key-dependent s-boxes. Being a Feistel cipher it has 16 rounds. Each round of the Blowfish algorithm has four steps. In the nth round, the left part of the block is XORed with the nth element in the subkey-array followed by passing it to the round function F. The output of function F is XORed with the right half of the initial block and then swapped. The round function F divides the 32-bit input into four 8-bit blocks that are then fed to 4 different S-Boxes. The output of the 1st and 2nd s-box is added and the result is XORed with the output from the 3rd s-box and again added to the output of the 4th s-box. It is one of the fastest algorithms for encryption [2].

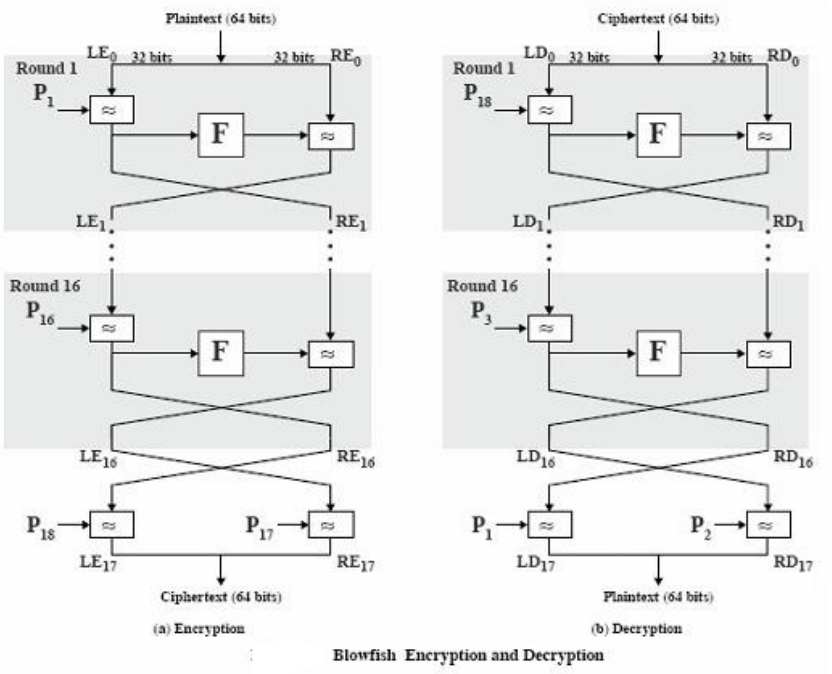


Fig 1. Blowfish Algorithm Encryption and Decryption [9]

AES - Advanced Encryption Standard

Rijndael is a Block Cipher developed by Belgian cryptographers, Vincent Rijmen and Joan Daemen that has been established as the Advanced Encryption Standard by the U.S. National Institute of Standards and Technology (NIST) in 2001 [7]. The AES has key lengths of 128, 192, 256 bits, and 10, 12, 14 rounds respectively. The 128 Bit key is expanded using AES Key Schedule into several subkeys depending on the number of rounds. In the beginning, an Initial Round Key is XORed to the input block. Then, for the first N-1 rounds, where N is the number of rounds, 4 Round Functions are applied on each block. The first-round function is Substitute Bytes where each byte is replaced with another according to a lookup table. Followed by ShiftRows, where the last three rows of the state are shifted cyclically a certain number of steps. MixColumns, a linear mixing operation that operates on the columns of the state, combining the four bytes in each column is applied to the block after ShiftRows. Lastly, AddRoundKey where each byte of the state is combined with a byte of the round key using bitwise xor is applied on the block. For the Nth round, i.e the last round all the above functions are applied except the Mixed Columns step [7]. It is one of the most widely used and secure encryption algorithms for data security. Even though it is slower than blowfish, it provides a higher level of data security [2]

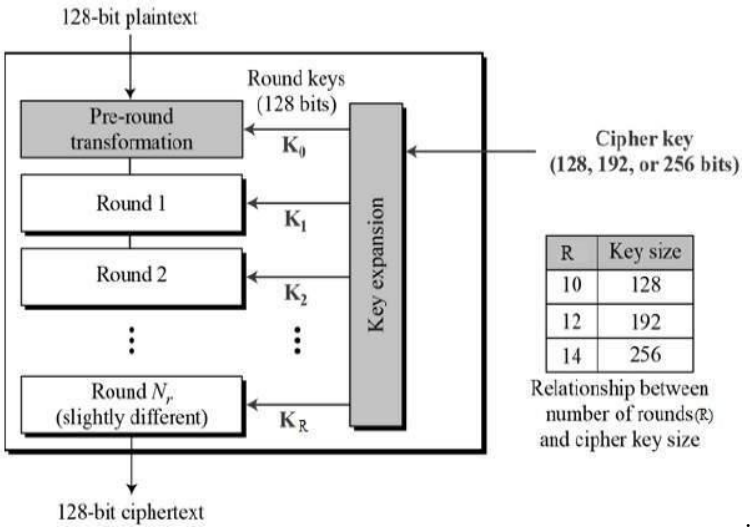


Fig 2. AES Algorithm Encryption and Decryption [10]

RSA Algorithm

Rivest–Shamir–Adleman (RSA) Algorithm is an asymmetric cryptographic algorithm that uses a Public Key, available to everyone on the network, to encrypt data and a Private Key, available to only the Sender and Receiver, for decryption. The keys are large prime numbers of lengths 1024 / 2048 / 3072 / 4096 Bits.

Two large prime numbers p and q are selected. The modulus n is calculated as, $n = p \times q$

Euler's Totient Function of n, $\varphi(n)$ is such that, $\varphi(n) = (p-1) \times (q-1)$

The Public key, e is selected such that e and $\varphi(n)$ are co-primes, i.e $\gcd(e, \varphi(n)) = 1$

The Private Key, d is calculated such that $(d \times e) \bmod \varphi(n) = 1$

Hence the Public Key pair is (e , n) and the Private Key Pair is (d , n).

Using RSA, encryption of the plaintext, M is done using the Public Key, e as:
Ciphertext, $C = M^e \bmod n$

The Ciphertext, C is decrypted using the Private Key, d as:
Plaintext, $M = C^d \bmod n$

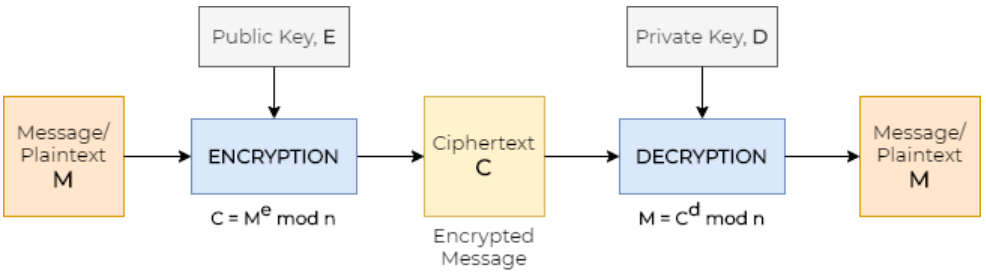


Fig 3. RSA Algorithm Encryption and Decryption

LSB Image Steganography

Least Significant Bit Steganography or LSB Steganography is the method of hiding secret data inside any form of digital media, here, Image. Images are made up of pixels, and the value of each pixel usually refers to the color of that particular pixel. In a grayscale image, these pixel values range from 0-255, 0 being black and 255 being white. In LSB Image Steganography, changing the last bit value of a pixel, won't have much of a visible change in the color. A cover image is used to embed the data in the Cover Image is converted to greyscale. The message is converted into binary. Each pixel of the image is traversed through, and for each pixel, initiate a temporary variable, temp. If the LSB of the Pixel Value and the message bit is the same, set temp as 0 and set temp as 1 otherwise. Update the output image pixel as image pixel value added with the temporary variable value, temp. This is done until the message is completely embedded

SHA-1 Hashing Function

Secure Hashing Algorithm is a one-way hash function that produces a condensed hash of the message, called the message digest. Any changes made to the message get reflected onto the message digest, that is if the message changes the message digest will change. This feature of SHA-1 is useful in the generation and verification of digital signatures, message authentication codes, generation of random numbers and bits. [8]

Design of Proposed System

Data Encryption using Blowfish, RSA & AES Encryption Algorithms in Cascading manner.

The System consists of three Encryption Layers, a Key Generator, and a List of Keys. The Key Generator generates the random n-bits Key depending on the Encryption Algorithm, while the List of Keys stores the Key Generated in each layer.

The plaintext P is first Encrypted using the Blowfish Algorithm with a 32 Bit / 64 Bit / 128 Bit Key, $K_{Blowfish}$. The Key $K_{Blowfish}$ is generated by the Key Generator and is used for Blowfish Encryption. It is then appended to the List of Keys, L. The Plaintext, P is encrypted to generate Ciphertext C_1.

$$C_1 = \text{Blowfish}(\text{Plaintext} = P , \text{Key} = K_{Blowfish})$$
$$L = [\,] \oplus K_{Blowfish}$$

The Ciphertext, C_1 is then encrypted using RSA Encryption with the 1024/2048 Bit Public Key, $K_{RSA\text{-}Public}$ generated by the Key Generator. A Private Key, $K_{RSA\text{-}Private}$, is also generated for Decryption. While the Public Key is used in Encryption, it is not stored in the List of Keys, L. The Private Key generated is appended to the List of Keys. C_1 is encrypted to generate Ciphertext C_2.

$$C_2 = \text{RSA}(\text{Plaintext} = C_1 , \text{Key} = K_{RSA\text{-}Public})$$
$$L = [K_{Blowfish}\,] \oplus K_{RSA\text{-}Private}$$

The Ciphertext, C_2 is then encrypted using AES-128 Encryption with the 128 Bit, K_{AES} generated by the Key Generator. The Key, K_{AES} generated is appended to the List of Keys, L. This Step gives the final encrypted ciphertext C.

$$\text{Ciphertext, } C = \text{AES}(\text{Plaintext} = C_2 , \text{Key} = K_{AES})$$
$$L = [K_{Blowfish} , K_{RSA\text{-}Private}\,] \oplus K_{AES}$$

The output of the system is the Ciphertext, C, and the list of keys L with all the keys.

$$\text{Ciphertext, } C$$
$$\text{List of Keys, } L = [K_{Blowfish} , K_{RSA\text{-}Private} , K_{AES}]$$

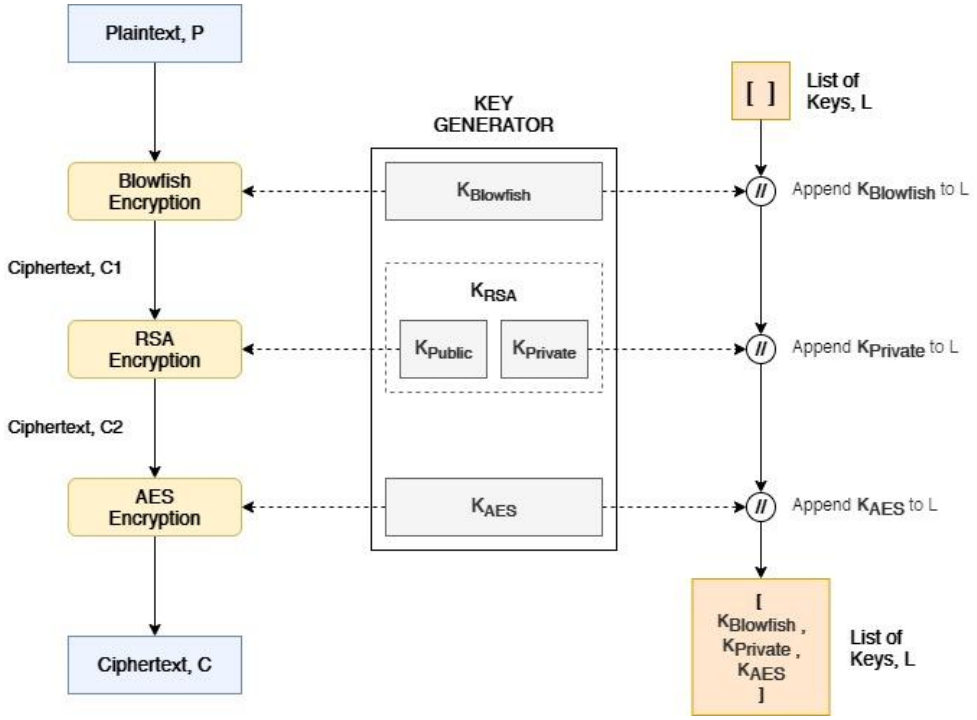


Fig 4. Data Encryption using Cascading Cryptographic Systems

Secure Key storage using encryption of Keys and LSB Steganography Image Embedding.

Using the proposed system, the Keys used for encryption at the various layers can be securely stored. The List of keys, L stores all the keys generated throughout the Data Encryption Process. Whenever the key for a particular Encryption Layer is generated, it is appended to the List of Keys, L.

In the system, the encryption layers are Blowfish, RSA, and AES, respectively, so the Keys used, are stored in the same order as:

$$L = [\ K_{Blowfish}\ ,\ K_{RSA\text{-}Private}\ ,\ K_{AES}\ ]$$

This List, L is then passed into a function that converts the list into a single string of keys separated by separators (x , * , /).

$$L_S = \text{Stringify(L, separator = 'x')} = K_{Blowfish}\ x\ K_{RSA\text{-}Private}\ x\ K_{AES}$$

The String, L_S is then encrypted using the AES Encryption Algorithm with a Key generated from user-input password. The user inputs a password, P_W which is hashed using SHA-1, and the first 16 Bits of the Hash is used as the key $K_{Password}$. The Key, $K_{Password}$ is used for the Encryption, generating the encrypted string $L_{S\text{-}Encrypted}$.

$$\text{Hashed Password, } H_P = SHA(P_W)$$

$$\text{Key, } K_{Password} = H_P[\ 0 : 16\ ]$$

$$L_{S\text{-}Encrypted} = AES(\ L_S\ ,\ K_{Password}\ )$$

This Encrypted string is then Embedded into a Cover Image using LSB Steganography, giving the embedded Stego Image.

$$\text{Stego Image} = \text{LSB\_Steganography(}\ L_{S\text{-}Encrypted}\ \text{, Cover Image)}$$

The Stego Image is transferred to the Receiver along with the Encrypted Data.

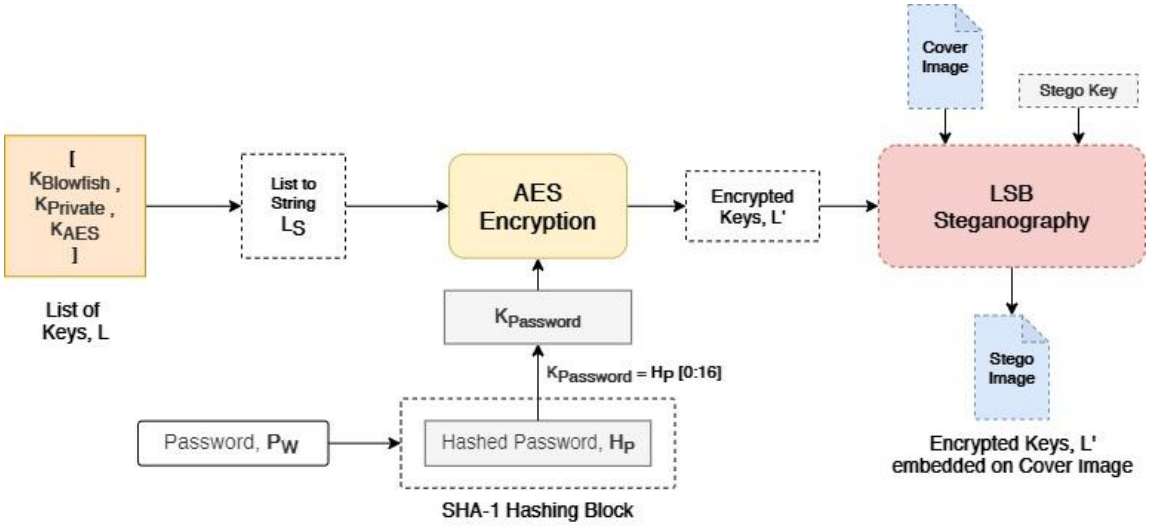


Fig 5. Secured Key Encryption and Storage in image using Steganography

Outcomes and Results

The Proposed Hybrid Crypto-system was implemented using Python and tested on a Windows PC with an Intel i3 processor and 4 GB of RAM. For demonstrating the Encryption of data, the following plaintext was encrypted. To encrypt the keys, **'enc2021'** was used as the password.

Plaintext :

Hello, World! This is 2021!

Ciphertext:

8afd0bbae83aff941a6c850d49a49bad5082f02bb6d9985c07efbab14c90dba02bc41f644553fa3b83e01b08f3b000d36ff82a32f9cc110bd2d30e710e20cd0aafa18f562a3cd58b6e8c39e14a88ec00c90949d43b07918f2d6519bad3894ca3c68adf8a79384922b353f8ebd1653cfa7eb894136a7066562f49624929fcea6cc65c809e7547a9cbe2f9c15444b9c4276798ace196932e73ade9abfc5ffa9e68646238de55d703d0694135353907c4e2beed80c9fdc0094a03ca0934db29844ea90c6f65006ed053a0d612c9b921c6f3c2a06fca7ad261e7e89fe6f3bb0f42519e6397f8c025284d6ad79c21351768b3f44c1792ca8848a8c67695d126d35aea2a142cc7d73ec7e2297e96fe17fdec9d

```
Enter Plaintext: Hello, World! This is 2021!
Enter Password: enc2021
Ciphertext:
8afd0bbae83aff941a6c850d49a49bad5082f02bb6d9985c07efbab14c90dba02bc41f644553fa3b83e01b08f3b000d36ff82a32f9cc110bd2d30e710e20cd0
aafa18f562a3cd58b6e8c39e14a88ec00c90949d43b07918f2d6519bad3894ca3c68adf8a79384922b353f8ebd1653cfa7eb894136a7066562f49624929fcea
6cc65c809e7547a9cbe2f9c15444b9c4276798ace196932e73ade9abfc5ffa9e68646238de55d703d0694135353907c4e2beed80c9fdc0094a03ca0934db298
44ea90c6f65006ed053a0d612c9b921c6f3c2a06fca7ad261e7e89fe6f3bb0f42519e6397f8c025284d6ad79c21351768b3f44c1792ca8848a8c67695d126d3
5aea2a142cc7d73ec7e2297e96fe17fdec9d
Encryption Complete!
Encryption Metrics:
Length of Plaintext   :  54
Length of Ciphertext  :  544
Length of Password    :  7
Encryption Time (sec) :  0:00:00.959421
```

Results show that the ciphertext is 10 times that of plaintext in length. It can be seen that the plaintext and ciphertext differ by a huge margin and are random, hence we can infer that the system can successfully encrypt the data to a form very different from the original plaintext.

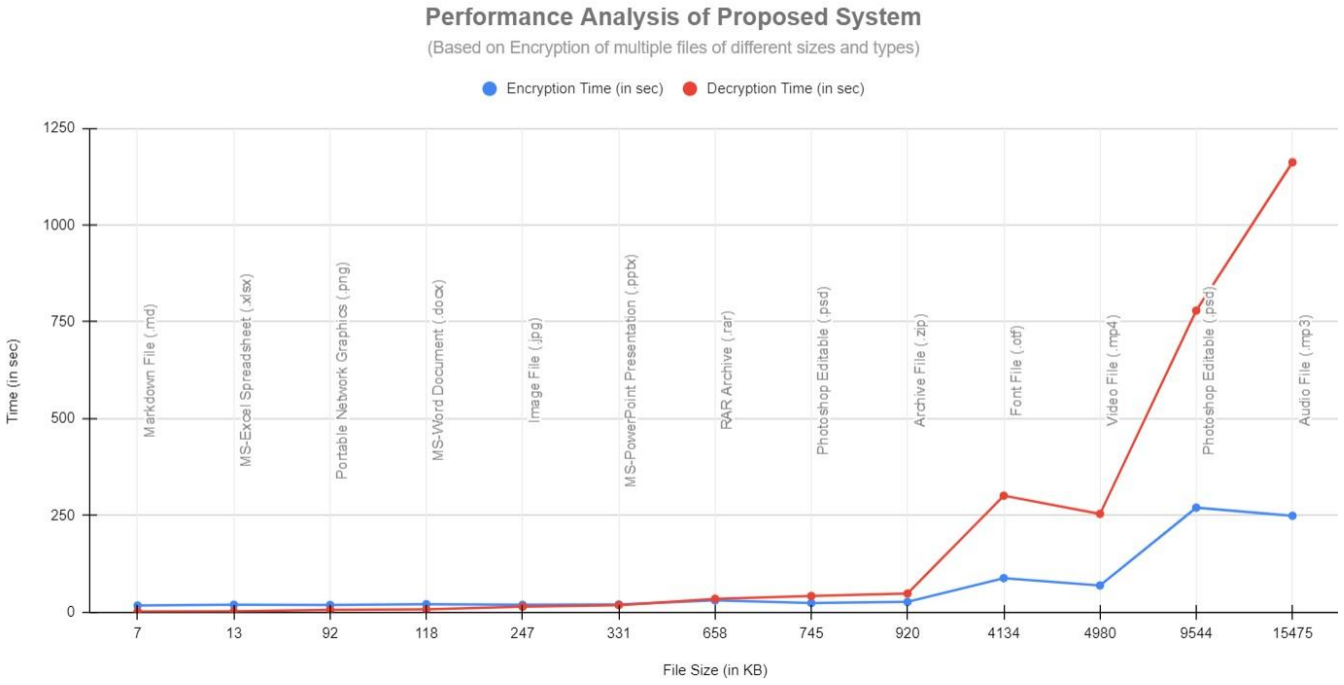
Performance Analysis of the Proposed System

For performance analysis Files of different sizes and types were encrypted using the proposed system with the same password, for every file. The password used was "**enc2021**".

File Type	Size (in KB)	Encryption Time (in sec)	Decryption Time (in sec)	Encryption Rate (in KB/sec)	Decryption Rate (in KB/sec)
Markdown File (.md)	7	17.24	1.81	0.4060324826	3.867403315
MS-Excel Spreadsheet (.xlsx)	13	19.19	2.14	0.6774361647	6.074766355
Portable Network Graphics (.png)	92	18.69	6.28	4.922418406	14.64968153
MS-Word Document (.docx)	118	20.75	7.31	5.686746988	16.14227086
Image File (.jpg)	247	19.06	14.26	12.9590766	17.32117812
MS-PowerPoint Presentation (.pptx)	331	20.09	18.25	16.47585864	18.1369863
RAR Archive (.rar)	658	30.37	34.73	21.66611788	18.94615606
Photoshop Editable (.psd)	745	23.96	41.92	31.09348915	17.77194656
Archive File (.zip)	920	26.65	48.1	34.52157598	19.12681913
Font File (.otf)	4134	88	301	46.97727273	13.73421927
Video File (.mp4)	4980	69	254	72.17391304	19.60629921
Photoshop Editable (.psd)	9544	270	779	35.34814815	12.25160462
Audio File (.mp3)	15475	249	1162	62.14859438	13.31755594

Table 1. Performance Analysis of proposed system

The results obtained showed that the crypto-system, when encrypting large-sized files, has a decryption time, significantly higher than encryption time. But when encrypting smaller-sized files the encryption time is slightly higher than the decryption time.



Graph 1. Performance Analysis of proposed system

Total Size of Files Encrypted: (in KB)	Average Encryption Time (in sec):	Average Decryption Time (in sec):	Average Encryption Rate (in KB/sec):	Average Decryption Rate (in KB/sec):
37264	67.08	205.45	26.54	14.69

On average, we observe that the Rate of Encryption is higher than the Rate of Decryption. Studies have shown that crypto-systems with higher decryption time take more time to break and hence are less susceptible to attacks and more secured [6]. Hence we can conclude that the proposed cryptosystem with high decryption time and rate, is highly secure and efficient in encrypting data.

Scope of Further Research and Improvements

The proposed system is very secure and robust. It has proven to encrypt data and ensure key security. While it is efficient and secure, it was also seen that the encrypted files are generally 2-3 times the size of the original file, hence the encrypted file takes up a significant amount of space to store. This drawback can be addressed by studying it further and making changes to the proposed system. Another improvement that can be made is by making the encryption and decryption time lesser. Further research on the proposed system can also be done by analyzing different order of combinations of the three algorithms used. A slightly different combination can also be studied by replacing one of the algorithms for improved performance.

Conclusion

The proposed cryptosystem uses a combination of symmetric and asymmetric cryptography to secure data. The system also introduces a sub-process to encrypt the keys used for encryption before embedding them in an image. The combination of Blowfish-RSA-AES has significantly improved the security and also ensured that the drawbacks of the standalone systems are addressed. The system also helps in improving security without the use of keys of larger lengths. We have also seen from the test results that the system is less susceptible to brute force attacks as the decryption time is significantly high [6]. The manyfold expansion of plaintext into ciphertext also helps in ensuring a high level of security. While the system successfully does its intended work, it still required minor improvements for larger adoption.

References

[1] Sandeshi, Sachithi & Priyanjana, W & Sajindra, Hirushan & Bandara, Hansi. (2020). “RSA in Communication.”

[2] Ghosh, Archisman. (2020). “Comparison of Encryption Algorithms: AES, Blowfish and Twofish for Security of Wireless Networks.” 10.13140/RG.2.2.31024.38401.

[3] V. kaul A. P Shaikh, “Enhanced Security Algorithm using Hybrid Encryption and ECC,” IOSR J. Comput. Eng., vol. 16, no. 3, pp. 80–85, 2014.

[4] Muin, Muhammad & Setyanto, Arief & Sudarmawan, & Santoso, Kartika. (2018). “Performance Comparison Between AES256-Blowfish and Blowfish-AES256 Combinations.” 137-141. 10.1109/ICITACEE.2018.8576929.

[5] Chitra Biswas Udayan Das Gupta Md. Mokammel Haque . “An Efficient Algorithm for Confidentiality, Integrity and Authentication Using Hybrid Cryptography and Steganography.“ 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)

[6] Inayatullah, Analisis Penerapan Algoritma MD5 Untuk Pengamanan Password, vol. 3, no. 3. 2007.

[7] Daemen, Joan & Rijmen, Vincent. (1998). The Block Cipher Rijndael. Lecture Notes in Computer Science - LNCS. 1820. 277-284. 10.1007/10721064_26.

[8] Leighton Johnson, Chapter 11 - "Security component fundamentals for assessment". Security Controls Evaluation, Testing, and Assessment Handbook (Second Edition). Academic Press, 2020, Pages 471-536, ISBN 9780128184271

[9] Kofahi, Najib. (2013). An Empirical Study to Compare the Performance of some Symmetric and Asymmetric Ciphers. International Journal of Security and Its Applications. 7. 1-16. 10.14257/ijsia.2013.7.5.01.

[10] Tutorials Point. Advanced Encryption Standard.