

A Triple Phase Hybrid Security Model for Cloud Storage Using Advanced Encryption Standard

Tashara Solomon, Yusuf Musa Malgwi, Molta Danlami Eli and Carroll Sermeje Pius

Department of Computer Science, Faculty of Physical Science Modibbo Adama University, Yola, Nigeria.

Corresponding author: sorasun02@gmail.com

Abstract

Cloud storage is an integral part of modern computing; however, the associated data security concerns have hindered its widespread adoption. This paper proposes a triple phase hybrid security model for cloud storage using Advanced Encryption Standard (AES) that can address these security concerns. In the first phase, the AES algorithm is used to encrypt the data using a shared key between the user and the cloud server. In the second phase, a data integrity verification algorithm is applied to guarantee that the data being stored in the cloud are not tampered with. Lastly, a transmission security layer is employed to protect data transmissions between the user and the cloud server. The data is encrypted using AES during the encryption process using 16-bit key. The encrypted data is inserted into a cover image during the steganography stage. The data is once more encrypted using AES and stored in the cloud during the hybrid phase. The effectiveness of the suggested model was assessed using a variety of security measures. The proposed triple phase hybrid security model is evaluated in an extensive security experiment. The results demonstrate that the proposed security model is able to effectively protect data stored in the cloud from threats such as unauthorized access, data manipulation and data leakage. Furthermore, the proposed model is also able to effectively minimize the transmission overhead and reduce the total computational cost required for data encryption and decryption.

Keywords: Advanced Encryption Standard, Computer Network, Cryptography, Information Security, Steganography, Cloud Storage, Hybrid Model.

Introduction

Data transfer and storage have become extremely important as a result of the increased use of technology, as evidenced by the adaption by people, businesses, and governments. Therefore, there is a requirement for greater information security. According to Ozighor and Izegebu (2020) security threats like Deliberate Software Attacks, Espionage or Trespass, Intrusion attack or Hacking, Spoofing, Man-in-the-Middle, Eavesdropping, Denial-of-Service attack, Malicious Software, and so on, can compromise the integrity, confidentiality, and availability of crucial and sensitive information transmitted over the insecure internet environment. Bank account statements, personal data, credit card details, trade secrets, official documents, etc. may be included in this crucial information. There are

numerous growing technologies, such as cloud computing and the Internet of Things (IoT), yet despite how beneficial they are, customers still have concerns about them.

Steganography was created as a result of researchers realizing that cryptography alone could not address the security issue because it was attackable. Steganography simply refers to writing a message with invisible ink that requires the use of another technology to make it visible. There are two types of image steganography, based on the domain: frequency domain-based methods and spatial domain-based methods (Biswas et al., 2019). In a frequency domain-based technique, the messages are stored in the transform coefficients after the images have first been translated to the frequency domain. In a spatial domain-based technique, messages are directly embedded in the intensity of picture pixels.

Steganography has been classified into multiple domains depending on the many types of covers since the cover might be text, image, audio, video, and more. In the instance of image steganography, the secret is hidden in the cover image.

Each of the aforementioned methods (steganography and cryptography) has a drawback to some degree. For instance, cryptographic algorithms can boost security by making calculations more difficult, but this would reduce performance (Hosam and Ahmad, 2019). Additionally, the mere sight of a scrambled message (ciphertext) might draw an observer's attention right away, signaling the importance of the message. Steganography also 'simply' masks a message's visual appearance; if an attacker discovers the presence of a message, he can easily decrypt it.

Information security has advanced to the point that it now integrates both steganography and cryptography to increase the effectiveness of data security, with a focus on maximizing resource efficiency while upholding excellent performance and data security. These two methods complement each other's shortcomings and are regularly combined to attain high levels of security (Ashok et al., 2010). The rising frequency of significant attacks on vital cyber networks is one of the world's greatest economic and security threats. In spite of the increasing number of research studies and solutions in the field of information security, there is a lack of efficient work that fully addresses the problems, necessitating ongoing effort. Prior studies have suggested merging steganographic and cryptographic methods to enhance message storage and transmission security. The finest cover option in picture steganography, however, guarantees the least visible stego image, ensuring the security of secret communication (Subhedar, 2021). When a huge message is embedded within a lower-dimensional image, there will be significant image distortion. However, putting little messages on images with bigger dimensionalities takes a lot of effort.

Statement of the Problem

One of the most severe economic and international security dangers the world faces is the growing frequency of serious attacks on critical cyber networks. Despite the growing number of research studies and solutions provided in the field of information security, there is no one effective work that addresses all of the challenges, necessitating a continual

effort. Numerous strategies have been proposed by researchers to deal with confidentiality, one of which being cryptography. This is due to the fact that all transactions must be entirely secret and encrypted from beginning to end. Data security is achieved through encryption, a process that turns data into a code (Ghosh, 2020).

Objectives of the Study

The objectives are to:

- i. Design a hybrid model of Advanced Encryption Standard (AES), Rivest, Shamir and Adleman (RSA) and Least Significant Bit (LSB).
- ii. Implement the model in using python.
- iii. Evaluate the efficiency of the proposed model in terms of encryption and decryption time, Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM).

This study aims to develop a triple-phase security model for cloud storage based on Advanced Encryption Standard. It covers design of the hybrid AES, Rivest, Shamir and Adleman (RSA) and Least Significant Bit (LSB), implementation the model using python language and evaluation in terms of encryption and decryption time, based on Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM). The study will basically look at the security of cloud data storage through cryptography and steganography.

Literature Review

In order to guarantee confidentiality, integrity, and availability, Information security, according to Nieves et al. (2017), is the safeguarding of data and information systems against unauthorized disclosure, disruption, access, use, modification, or destruction. The most crucial component of communication security is cryptography, which also serves as the foundation for computer security. Through the use of codes, cryptography protects data and communications while allowing only those who require it to read and process it to do so. A "Key" is a piece of information that an algorithm uses to change data, and it is a key notion in cryptography. Its objective is to obscure this information and limit access to those who possess the necessary keys to decipher it (Hassan & Hijazi, 2016). Without the secret key, neither you nor an adversary can decrypt or understand any of the encrypted data. According to Khalifa et al. (2004), private key cryptography, sometimes referred to as symmetric key cryptography, requires the sender and receiver to share a key, which is a secret bit of information used to encrypt or decrypt a message. Private keys are frequently smaller and compute more quickly since they contain fewer pieces of information. The difficulty of a brute-force attack, which involves testing a large number of keys until the correct one is found, is dependent on the complexity and length of the private key (Madura, & Ngo, 2014). Private key cryptography encrypts and decrypts data using a single private key.

Public key cryptography is a type of encryption that employs two keys—a public key and a private key—that are mathematically related but not the same (Joshi et al., 2022). Anyone can use the public key to encrypt data and send it to the recipient; however, only the holder of the private key will be able to decrypt. Steganography, according to Zagade and Bhosale (2014), is the practice of concealing the use of one transmission channel in order to achieve secret communication. Steganography's objective is to hide communications inside of other messages such that no opponent can figure out that there is a second message present. The future of Internet security and privacy in open networks like the Internet depends heavily on steganographic technologies. Steganographic research is motivated by the need for complete privacy in an open systems context as well as the weakness of standalone cryptography methods. Information can be concealed using Steganography through the use of audio, video, text, and image carriers. Based on two cryptographic methods, Santoso (2021) suggested a hybrid strategy for safe data transmission. The procedure depended on encrypting the intended message using the Hill Cipher. Due to RSA's use of big prime numbers for security purposes, the Hill Cipher key was then encrypted in order to handle key storage and administration. Accordingly, the output of the system is the ciphertext (encrypted with the Hill Cipher), the Hill Cipher key (encrypted with RSA), and then the RSA keys. An interface for managing all of these operations was made using Visual Studio. Similar plaintext blocks are encrypted into similar ciphertext blocks, which is a key drawback of the Hill Cipher technique.

Using AES and DES to encrypt data that will be uploaded to the cloud, Kumar et al. (2020) introduced a hybrid cryptographic technique for protecting cloud data. Data was first encrypted using the AES, which created a ciphertext (C₁) that was utilized as input by the DES, which created a second ciphertext (C₂) that was later uploaded to the cloud. To finish the project, java programming was employed. They discovered their hybrid offered more security. Though security has advanced, it's crucial to remember that memory and time complexity have also grown.

For the security of data (picture, text, audio, and video), Tiwari et al. (2021) conducted a comparative examination of various Steganography algorithms. The studied works demonstrate that concealing data boosts its security and that encrypting data alone is not sufficient. According to them, LSB offers an improved concealment strategy that is hard to decipher quickly. To increase communication security, Hossen et al. (2020) suggested combining the DCT steganography algorithm with two cryptographic methods (RC5 and AES). The message is encrypted twice, once with RC5 and once with AES (both with 128-bit keys); this adds to the complexity of the overall strategy because it combines the time needed for both types of encryption. The DCT, which has significant picture distortion, is then used to conceal the encrypted message. A method for safeguarding data transfer using cryptographic and steganographic techniques was proposed by Manjula Shenoy and Shaikh (2019). This method encrypts the message to be sent using the Rivest-Shamir-Adleman (RSA) algorithm. Using the Least Significant Bit method (LSB), the ciphertext is

then concealed behind a web image. The receiver is now supplied the image's consistent resource location, and they will utilize the LSB to reveal the ciphertext and subsequently the RSA key to decrypt it. However, this method has serious security flaws because RSA is ineffective for lengthy texts. In addition, the core problems with RSA include speed and the use of subpar or small prime numbers.

The combined use of encryption and steganography affords messages a higher level of security, as demonstrated by all the studies discussed above. However, the procedure as a whole is significantly impacted by the use of cryptography and steganographic approaches. The management of encryption keys is especially crucial because key disclosure compromises not just current communications and storage but also future ones. In order to address these issues, this work proposes the combination of two powerful cryptographic approaches and also offers a way to secure the keys using steganography.

The Proposed Work

This paper aims to develop a triple-phase security model for cloud storage based on Advanced Encryption Standard.

Methodology

Developing a Triple Phase Hybrid Security Model for Cloud Storage using the Advanced Encryption Standard (AES) requires careful planning and consideration of various components.

- The desired level of security, compliance standards, performance benchmarks, and scalability needs for cloud storage identified.
- The potential threats and vulnerabilities in the cloud storage environment were identified to help in designing appropriate security measures.
- The appropriate components for each phase of the hybrid security model include encryption algorithms, key management systems, authentication mechanisms, access control policies, and intrusion detection/prevention systems.
- The AES key size 16-bits based on the desired security level and performance requirements was determined.
- The secure communication protocols to protect data in transit was implemented.
- the access control mechanism and key management system to handle the encryption keys securely were developed and rigorous security testing, including penetration testing and vulnerability assessments, to identify any weaknesses in the model was conducted.

Architecture of the Proposed Hybrid Model

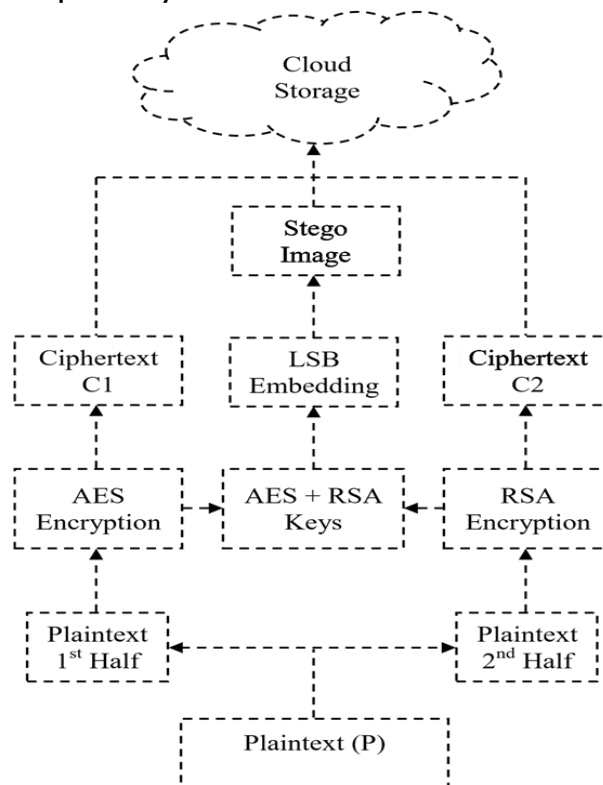


Figure 1: visualizing the Architecture of the proposed Hybrid model

The python programming language was used for development purpose (with Jupyter Notebook as the IDE). A hybrid model combining the Advanced Encryption Standard (AES), Rivest-Shamir-Adleman (RSA), and Least Significant Bit (LSB) techniques can be used for encryption and steganography purposes. The hybrid model's potential operation is described in detail as follows:

- Create an RSA key pair, which consists of a public key and a private key. The AES symmetric key will be encrypted with the public key and decrypted with the private key.
- AES Encryption: Using the AES technique and a symmetric key that is created at random, encrypt the plaintext data. The original data will become unintelligible cipher text as a result of this process.
- RSA Encryption: Use the recipient's public RSA key to encrypt the AES symmetric key. As only the recipient who has the associated private key may decrypt the AES key, this phase provides secure key exchange.
- LSB Steganography: Choose an image to serve as the data carrier for the AES encryption. Use least significant bit (LSB) steganography to hide the RSA-encrypted AES key in the image's pixel values. With minimal visual impact, this technique conceals the encrypted data within the image.

- Send the image to the appropriate recipient after having it transformed via steganography. The AES-encrypted data is embedded within what appears to be a conventional image.
- Decryption Procedure: After receiving the steganographic image, the recipient uses LSB steganography techniques to decrypt the AES data.
- RSA Decryption: Using the recipient's personal RSA key, decrypt the RSA-encrypted AES key that was retrieved. By using this method, the original AES symmetric key is obtained.

Using the AES symmetric key that has been decrypted, decrypt the cipher text. Now, the recipient can get the original plaintext information. This hybrid paradigm can offer both confidentiality and obscurity in data transport and storage by fusing RSA for safe key exchange, AES for robust encryption, and LSB steganography to conceal the encrypted data.

Performance Metrics for Evaluation of Proposed Model

The importance of determining the degree of distortion on intended photographs cannot be emphasized because an attacker may use this to infer the presence of a message. The measures used were the encryption and decryption times, the peak signal to noise ratio, and the structured similarity indexing approach. Calculating a program's execution time is particularly helpful since it aids researchers and programmers in optimization, which is important given how valuable time is. One of the formulas to use is

$$\text{ExecTime} = \text{StopTime} - \text{StartTime} \quad (1)$$

where StopTime is the time the task ends executing and StartTime is the time the task begins executing, to determine the encryption and decryption times of a given task/algorithm.

The time can be obtained in several ways for various programming languages. Peak signal to noise ratio (PSNR) is a measure of the relationship between the maximum signal power and the noise power that distorts the signal's quality of representation. The most used quality assessment metric to gauge the caliber of reconstructed images is the PSNR, which provides an approximation of human perception of reconstruction quality. The metric is shown as follows:

$$PSNR = 20 \log_{10} [MAX - MSE_i] \quad (2)$$

Where MAX_i refers to the maximum possible pixel value of the image and MSE is the Mean Squared Error between the modified and original image represented as:

$$MSE = \frac{1}{M * N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [I(i, j) - K(i, j)]^2 \quad (3)$$

where M represent the width, N the height of the images, I(i, j) the modified image and K(i, j) the original image. The row and column pixels of both the original and modified images are represented as i and j.

Structured similarity indexing method (SSIM): The SSIM is one of the most widely used methods to measure the similarity between two images. This metric quantifies the reduction in image quality caused by data compression or transmission losses, given by:

$$SSIM = \frac{(2\mu_I\mu_{De} + C_{I_1})(2\sigma_{IDe} + C_{I_2})}{(\mu_I^2 + \mu_{De}^2 + C_{I_1})(\sigma_I^2 + \sigma_{De}^2 + C_{I_2})} \quad (4)$$

where μ_I denotes the average of the input (I) and μ_{De} that of the decrypted (De) images. The variance of I and De are σ_I^2 and σ_{De}^2 respectively. σ_{IDe} signifies the covariance of the values I and De, while C_{I_1} and C_{I_2} represent the regularization.

The images used for the experiment to hide the merged keys are described in Table 1. They are benchmark images used for steganography and the source of the images is the DIV2K dataset.

Table 1: Description of Steganography Images

S/No.	Image Name	Type	Size
1.	Babylon	PNG	3.48Mb
2.	Lenna	PNG	462Kb
3.	Pepper	PNG	280Kb

Implementation and Analysis

Implementation

The system was implemented using the Jupyter Notebook IDE. The components of the python code, resources, and supporting files that were developed and used to implement the cryptographic and steganographic systems are further explained in the subsequent subsections.

```
In [1]: 1 import hashlib
        2 from Crypto import Random
        3 from Crypto.Cipher import AES
        4 from base64 import b64encode, b64decode
        5
        6 import random
        7
        8 from datetime import datetime
        9
       10 import cv2, os, shutil, glob, numpy as np
       11 import math, xlwt
       12 from PIL import Image
       13 from skimage.measure import compare_ssim as ssim
       14 from skimage.measure import compare_psnr as psnr
       15 import matplotlib.pyplot as plt
       16 %matplotlib inline
       17 import matplotlib.image as img
```

Figure 1: Importing of libraries

The first four (4) lines importations were libraries needed to achieve AES encryption, the random importation (line 6) was used to achieve RSA encryption, the datetime import (line 8) was used for analysis of execution time and the last two importations (line 10 and 11) were used to achieve LSB steganography.

```
In [2]: 1 def bin_conv(str_data):
        2     res = ''.join(format(i, '08b') for i in bytearray(str_data, encoding='utf-8'))
        3     return res
```

Figure 2: Binary conversion

Here, a method named bin_conv was created, which accepts String data and returns a binary data after successful conversion.

```
In [3]: 1 def my_split(data):
        2     split1 = ""
        3     split2 = ""
        4     d_len = len(data)
        5     d_len2 = round(d_len/2)
        6
        7     for x in range(d_len2):
        8         split1 = split1 + data[x]
        9     for x in range(d_len2, d_len):
        10         split2 += data[x]
        11     return split1, split2
```

Figure 3: Data Splitting

A method called my-split was created to split the converted binary data into halves. The data needs to be divided into halves – the first half is stored in split1 and the second half in split2. After successful splitting, the method returns two arrays (split1 and split2).

Result and Analysis

```
In [4]: 1 class AESCipher(object):
2     def __init__(self, key):
3         self.block_size = AES.block_size
4         self.key = hashlib.sha256(key.encode()).digest()
5
6     #Pad
7     def __pad(self, plain_text):
8         number_of_bytes_to_pad = self.block_size - len(plain_text) % self.block_size
9         ascii_string = chr(number_of_bytes_to_pad)
10        padding_str = number_of_bytes_to_pad * ascii_string
11        padded_plain_text = plain_text + padding_str
12        return padded_plain_text
13
14    #Unpad
15    @staticmethod
16    def __unpad(plain_text):
17        last_character = plain_text[len(plain_text) - 1:]
18        bytes_to_remove = ord(last_character)
19        return plain_text[:-bytes_to_remove]
20
21    #Encrypt
22    def encrypt(self, plain_text):
23        plain_text = self.__pad(plain_text)
24        iv = Random.new().read(self.block_size)
25        cipher = AES.new(self.key, AES.MODE_CBC, iv)
26        encrypted_text = cipher.encrypt(plain_text.encode())
27        enc_key = "hardlockencrypts"
28
29        return b64encode(iv + encrypted_text).decode("utf-8"), enc_key
30
31    #Decrypt
32    def decrypt(self, encrypted_text, aes_key):
33        encrypted_text = b64decode(encrypted_text)
34        iv = encrypted_text[:self.block_size]
35        cipher = AES.new(self.key, AES.MODE_CBC, iv)
36        plain_text = cipher.decrypt(encrypted_text[self.block_size:]).decode("utf-8")
37        return self.__unpad(plain_text)
```

Figure 4: AES Encryption

Figure 4 shows the AES Class consisting of five methods namely: __init__, __pad, __unpad, encrypt and decrypt. However, we are looking at the encrypt method which accepts the plaintext data and returns a ciphertext and a 16-bit key used for the encryption as specified within the method (which can be changed to any other 16-bit key).

```
85
86 def rsa_encrypt(pk, plaintext):
87     # Unpack the key into it's components
88     key, n = pk
89     # Convert each letter in the plaintext to numbers based on the character using a^b mod m
90     cipher = [pow(ord(char), key, n) for char in plaintext]
91
92     # Return the array of bytes
93     return cipher
94
95
96 def rsa_decrypt(pk, ciphertext):
97     # Unpack the key into its components
98     key, n = pk
99     # Generate the plaintext based on the ciphertext and key using a^b mod m
100    aux = [str(pow(char, key, n)) for char in ciphertext]
101    # Return the array of bytes as a string
102    plain = [chr(int(char2)) for char2 in aux]
103    return ''.join(plain)
104
105 public, private = generate_key_pair(p, q)
106 rsa_encrypted_msg = rsa_encrypt(public, data)
107
108 return rsa_encrypted_msg, public, private
109
```

Figure 5: RSA Encryption

The RSA encryption requires several other operations which were modeled as methods – gcd (greatest common divisor), multiplicative_inverse, is_prime, generate_key _pair, rsa_encrypt and rsa_decrypt. Because of the plenty number of lines, Figure 5 only shows a snapshot of the rsa_encrypt and rsa_decrypt which are the core. The rsa_encrypt was used to encrypt the second half of the data and returned a ciphertext (cipher2).

```
In [6]: 1 def merge(data1, data2):
        2     merged_data = data1 + data2
        3     return merged_data
```

Figure 6: Merge Data

A method named merge was created, which takes as input the two encryption keys (AES and RSA keys) and returns as a single text the merged keys which will then be hidden in a cover image.

```
In [7]: 1 class LSB():
        2
        3     # Encoding Part
        4     def encode_image(self, img, msg):
        5         length = len(msg)
        6         encoded = img.copy()
        7         width, height = img.size
        8         index = 0
        9         for row in range(height):
        10             for col in range(width):
        11                 if img.mode != 'RGB':
        12                     r, g, b, a = img.getpixel((col, row))
        13                 elif img.mode == 'RGB':
        14                     r, g, b = img.getpixel((col, row))
        15
        16                 # first value is Length of msg
        17                 if row == 0 and col == 0 and index < length:
        18                     asc = length
        19                 elif index <= length:
        20                     c = msg[index - 1]
        21                     asc = ord(c)
        22                 else:
        23                     asc = b
        24                 encoded.putpixel((col, row), (r, g, asc))
        25                 index += 1
        26             return encoded
        27
        28     # Decoding Part
        29     def decode_image(self, img):
        30         width, height = img.size
        31         msg = ""
        32         index = 0
        33         for row in range(height):
        34             for col in range(width):
        35                 if img.mode != 'RGB':
        36                     r, g, b, a = img.getpixel((col, row))
        37                 elif img.mode == 'RGB':
        38                     r, g, b = img.getpixel((col, row))
        39
        40                 # first pixel r value is Length of message
        41                 if row == 0 and col == 0:
        42                     length = b
        43                 elif index <= length:
        44                     msg += chr(b)
        45                 index += 1
        46         #lsb_decoded_image_file = "lsb_" + original_image_file
        47         return msg
```

Figure 7: LSB Steganography

The LSB encode method as shown in Figure 6, takes in a cover image and the message (in this case the merged keys) to hide and returns a stego image. While the decode method accepts as input a stego image and returns the hidden merged keys.

Encryption time: The experimental result in Table 2 shows that as the size of the data increases, the time taken to encrypt increases also (with RSA encryption on the high which is expected as shown in the literature review).

Table 2: AES and RSA Encryption Time

Round	Data Size	AES Encryption	RSA Encryption	Total
1	3Kb	0.011994	0.023983	0.035977
2	9Kb	0.033977	0.071257	0.105234
3	14Kb	0.039316	0.118178	0.157494

Table 3: AES and RSA Decryption Time

Round	Data Size	AES Decryption	RSA Decryption	Total
1	21Kb	0.011995	0.45987	0.47187
2	77Kb	0.033979	0.51754	0.55152
3	14Kb	0.039319	0.78664	0.82596

Decryption time: It can be noticed from Table 3 that the increase in decryption time for AES is negligible compared to the encryption, because encryption and decryption time for AES is almost the same. However, that is not the case with RSA (as it takes longer time in decryption than in encryption).

LSB results: The result obtained from hiding of the merged keys in the developed model is the input image (cover image), the output image (stego image), the PSNR values and SSIM values.

Table 4: Pictorial Representation of Cover and Stego Images



Table 5: PSNR and SSIM Values of Images used

S/No	Image	PSNR	SSIM
1.	Babylon	62.3967 dB	1.0000
2.	Lenna	59.6601 dB	1.0000
3.	Pepper	59.6665 dB	1.0000

From the results in the Table 5, it can be seen that PSNR value for Babylon is very high, this is because not much of the Least Significant bits were replaced (as it has a very large size compared to Lenna and Pepper). It can also be seen from Table 4.4 and then the SSIM values that there seem to be no change in the alteration of the images (this is because, SSIM works just like the human eyes).

Conclusion

This research successfully developed a new model for securing cloud data that employs two different cryptographic techniques (AES and RSA) and a steganographic technique (LSB). By utilizing the use of these two cryptographic techniques, the complexity of breaking the system by an attacker is increased. Furthermore, the keys are not openly available which is also a way of denying attackers access to the data. From the results presented, it can be

noticed that even though the time taken to decrypt is higher than the encryption time, it is still negligible as the user does not have to wait for much seconds to have his/her data encrypted or decrypted. The result of hiding our keys in the image also showed good values for both PSNR and SSIM, meaning that an attacker is not likely to suspect that the image used is carrying any message. In all, the study has found a way of securing data before being uploaded to the cloud and this method is computationally inexpensive. This is the major contribution of the work – encrypting plaintext data and then hiding the keys in a cover image. In the future, this study can be enhanced and expanded to the security of other kinds of data security like hybrid data transmission and steganography techniques for data security.

Recommendations

Based on the result of this study we recommend that:

- Organizations dealing with a significant amount of sensitive data, such as financial institutions, healthcare providers, or government agencies, may opt for this level of security to protect their valuable information.
- Companies offering cloud storage services can implement this Model to bolster the security of their platforms and gain a competitive edge by providing enhanced data protection to their customers.
- Organizations involved in research and development of cutting-edge technologies may employ this security model to safeguard intellectual property and confidential data.

References

- Ozighor, E. R., & Izegbu, I. (2020). Information Protection against Security Threats in an Insecure Environment using Cryptography and Steganography. *GSI*, 8(5).
- Ghosh, A. (2020). Comparison of Encryption Algorithms: AES, Blowfish and Twofish for Security of Wireless Networks. *International Research Journal of Engineering and Technology (IRJET)*, June, 4656–4659.
- Biswas, C., Gupta, U. D., & Haque, M. M. (2019, February). An efficient algorithm for confidentiality, integrity and authentication using hybrid cryptography and steganography. In *2019 international conference on electrical, computer and communication engineering (ECCE)* (pp. 1-5). IEEE.
- Hosam, O., & Ahmad, M. H. (2019). Hybrid design for cloud data security using combination of AES, ECC and LSB steganography. *International Journal of Computational Science and Engineering*, 19(2), 153-161.
- Ashok, J., Raju, Y., Munishankaraiah, S., & Srinivas, K. (2010). Steganography: an overview. *International Journal of Engineering Science and Technology*, 2(10), 5985-5992.
- Subhedar, M. S. (2021). Cover selection technique for secure transform domain image steganography. *Iran Journal of Computer Science*, 1-12.
- Nieves, M., Dempsey, K., & Pillitteri, V. Y. (2017). An introduction to information security. *NIST special publication*, 800(12), 101.
- Hassan, N. A., & Hijazi, R. (2016). *Data hiding techniques in Windows OS: a practical approach to investigation and defense*. Syngress.

- Khalifa, O. O., Islam, M. R., Khan, S., & Shebani, M. S. (2004). Communications cryptography, RF and Microwave Conference. In *RFM 2004. Proceedings*.
- Madura, J., & Ngo, T. (2014). Private information leakages and informed trading returns of tech target firms. *The Journal of High Technology Management Research*, 25(1), 36-53.
- Joshi, R., Bairwa, A. K., Soni, V., & Joshi, S. (2022, January). Data Security Using Multiple Image Steganography and Hybrid Data Encryption Techniques. In *2022 International Conference for Advancement in Technology (ICONAT)* (pp. 1-7). IEEE.
- Zagade, S., & Bhosale, S. (2014). Secret data hiding in images by using DWT Techniques. *International Journal of Engineering and Advanced Technology*, 3(5).
- Santoso, Y. S. (2021). Message Security Using a Combination of Hill Cipher and RSA Algorithms. *Jurnal Matematika Dan Ilmu Pengetahuan Alam LLDikti Wilayah 1 (JUMPA)*, 1(1), 20-28.
- Kumar, A., Jain, V., & Yadav, A. (2020, February). A new approach for security in cloud data storage for IOT applications using hybrid cryptography technique. In *2020 international conference on power electronics & IoT applications in renewable energy and its control (PARC)* (pp. 514-517). IEEE.
- Tiwari, A., Shankar, G., & Jain, B. B. (2021). Comparative Analysis of Different Steganography Technique for Image Security. *International Journal of Engineering Trends and Applications (IJETA)*, 8(2), 6-9.
- Hossen, M. S., Islam, M. A., Khatun, T., Hossain, S., & Rahman, M. M. (2020, September). A new approach to hiding data in the images using steganography techniques based on AES and RC5 algorithm cryptosystem. In *2020 International Conference on Smart Electronics and Communication (ICOSEC)* (pp. 676-681). IEEE.
- Shenoy, K. M., & Shaikh, S. G. (2019, July). An Approach to secure data transmission through the use of cryptography and steganography. In *2019 International Conference on Communication and Electronics Systems (ICCES)* (pp. 1039-1043). IEEE.