Techniques for Secure Data Transfer

Jeffery Gardner Jr.
Department of Computer Science, Grambling State University, USA

***Abstract* —** The Advanced Encryption Standard better known as the AES algorithm is a symmetric (uses the same key to encrypt and decrypt) cryptographic technique used in most of today's classified and unclassified data transfers. The AES algorithm provides data transfers with layers of security through its mathematical complexity. Alongside this technique of encryption, the act of concealing data within different objects is now becoming an essential component in the art of secure data transfer. This method of hiding secret messages within a file type is known as Steganography. Even though these two elements of secure data transfer differ, they both share the same objective namely to protect the integrity of the data. This paper provides an explanation of Steganography and AES algorithm and how they can be used together to enhance the security of data. The experiments that this paper demonstrates used the AES-256 algorithm.

KEYWORDS
Advanced Encryption Standard, steganography, algorithm, S-Box, cryptography.

1. INTRODUCTION

The Advance Encryption Standard algorithm (AES) is the standard encryption technique for today's classified and unclassified data transfers. The AES algorithm utilizes the Rijndael algorithm [1] that is a symmetric block cipher that operates on blocks of 128 bits giving by the required standard. As a standard, three different key lengths supports AES. They are AES-128, AES-192, and AES-256. The security of the AES algorithm heavily relies on the mathematical complexity of the five different layers of the giving components. The layers of the AES algorithm include the Key Expansion, Substitution Bytes, Shift Rows, Mixed Columns and Add Round Key. These layers will provide the AES algorithm with the necessary protection for secure data transfer.

Steganography has existed in the "secure transfer" community ever since 440 B.C., where Greek Kings shaved the head of a villager and placed important information on his/her scalp. The villager was then sent back to the village after his/her hair had grown back. With this method in mind, steganography can easily be described as providing security for sensitive data through the means of obscurity.

Steganography compared to cryptography poses a significant difference in objective. Encryption provides communication with security through the use of scramble letters and symbols. With cryptography, any adversary can clearly see that the transmitted message is encrypted and can be passed along for further cryptanalysis or can perform actions that can hinder the transmitted message without the secret key.

This paper will demonstrate how to encrypt data using the AES algorithm and utilizing steganography to embed information in a media file to provide obscurity for elementary data transfer. Section 2 explains the design of the AES algorithm and Section 3 discusses the previously known attacks of AES algorithm. A section 4 describes how steganography is used. Section 5 discusses related works. Section 6 will discuss the application of how the AES encryption and steganography used together for an elementary data transfer.

2. ADVANCED ENCRYPTION STANDARD

The AES algorithm is not only equipped for software purposes but also with providing security to hardware devices. The AES algorithm performs bitwise operations, making encryption for hardware efficient.

	Nk	Nb	Nr
AES 128	4	4	10
AES 192	6	4	12
AES 256	8	4	14

***Figure 1**:*

Figure 1 displays the number of rounds that are generated by each particular key length provided by the AES standard, where

Nk is the number of 32-bit words generated from the key length. These words are known as the sub-keys.

Nb represents the number of 32-bit words, made from the state (the AES standard operates on ***Nb*** = 4).

Nr is the number of rounds that are utilized for the given key length.

The five components that are responsible for the robustness of the AES algorithm are Key Expansion, Substitution Bytes, Shift Rows, Mix Columns, and Add Key Round. In the AES algorithm, each input value represents in the polynomial form in the $GF(2^8)$.

Key Expansion is responsible for generating the key schedule and is processed before encrypt and decrypt ciphers. In this process sub-keys are created. For the experiments in this paper, the AES-256 was utilized to test the highest level of security. Because of the AES 256, there were 60 words generated from the key expansion. Each key length of the AES encryption uses the formula

in equation (1) to produce the proper number of sub-words.

$$Nb\ (Nr\text{-}1) \qquad (1)$$

The substitution bytes are calculated using a pre-computed table known as the S-Box. For testing purposes, the S-Box was developed using C-programming language that was capable of accessing the hardware that derived S-Box values, which sits at the core of the security in the AES algorithm. The process accomplished to obtain a sound understanding of the importance of this component and how values were derived. In developing this individual component, the S-Box is efficient and provides non-linearity and performs a one-for-substitution of an input byte value. To calculate the substitution byte of the data value, the S-Box uses two transformation methods known as the modular inverse and the affine transformation. The modular inverse is responsible for finding the multiplicative inverse of the giving input byte by utilizing the extended Euclidean algorithm.

$$\{out\ byte\} = \begin{bmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{bmatrix} \{in\ byte\} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

***Figure 2**: Structure of affine transformation matrix operation*

After the inverse of the giving input value has been obtained, it was then sent through the affine transformation where the S-Box value is the result. Figure 2 displays the structure of affine transformation matrix operation. The affine transformation is accomplished by using the formula {out byte} = M {in byte} XOR {v} where *M* is a pre-determined matrix and *v* is the vector of the multiplicative inverse byte.

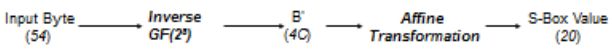

Figure 3: multiplicative inverse and affine transformation

Figure 3 demonstrates the use of the multiplicative inverse and affine transformation to obtain an S-Box value. In figure 3, the hexadecimal value of {54} was the input byte. The data byte of {54} was processed through the multiplicative inverse transformation finding a value that congruent to:

$$1 \bmod x^8+x^4+x^3+x+1 \qquad (2)$$

In this operation shown in equation (2), the Galois Field used to represent the polynomial form for the input byte. We represent these values in the $GF(2^8)$ which is the standard field for the AES algorithm. Using the irreducible polynomial $x^8+x^4+x^3+x+1$ with a degree of 8 that is provided by the norm is a modulo that corresponds to the multiplication of two polynomials. The value *B'* is the result of the multiplicative inverse. The result of this transformation is the hexadecimal value of {4C}. Finally, the multiplicative inverse result was used to obtain the S-Box value in the affine transformation. In this transformation, a simple matrix multiplication is applied using a pre-determined 4 x 4 matrix found in figure 2 and the multiplicative inverse result. The result of these two transformations is the S-Box value.

The inverse substitution byte function operates similar to the forward substitution byte function found in the encryption process providing non-linearity between any two values. However, the inverse substitution method utilizes the affine transformation and multiplicative inverse functions in reverse order. The Affine Transformation is the lead off transformation in this operation and applied first by using the inverse Affine Transformation matrix as in Figure 4.

$$\{out\ byte\} = \begin{bmatrix} 0&0&1&0&0&1&0&1 \\ 1&0&0&1&0&0&1&0 \\ 0&1&0&0&1&0&0&1 \\ 1&0&1&0&0&1&0&0 \\ 0&1&0&1&0&0&1&0 \\ 0&0&1&0&1&0&0&1 \\ 1&0&0&1&0&1&0&0 \\ 0&1&0&0&1&0&1&0 \end{bmatrix} \{in\ byte\} \oplus \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

***Figure 4**: inverse Affine Transformation matrix*

Figure 5 shows the result of this operation is the inverse of the original input value. The value obtained from the affine transformation is then passed to the multiplicative inverse function to obtain the original value.

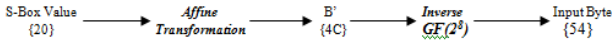

Figure 5: Inverse of the input operation

In this project, experiments took place that tested the speeds between the S-Box and inverse S-Box being accessed by the pre-computed tables as used in majority of AES programs and the hardcode of the various steps in deriving the S-Box and Inverse S-Box values. After conducting ten experiments it was proven that the hardcode of the S-Box and inverse S-Box operations took on average about 7.918 seconds. With the precomputed tables, this same operation took on average about 4.034 seconds. The results of this experiment conveyed that using the per-computed tables is on average 3.839 seconds faster than the hardcoded forward and inverse S-Box operation.

The Shifts Rows component of the AES algorithm is the second procedure of the encryption process and is responsible for shifting each row by a certain number of

spaces to the left.

Figure 6 displays the basic transformations of the shift rows function. The first row is not shifted at all, the second row is shifted one space to the left, the third row is shifted two spaces to the left, and the fourth row is shifted three spaces to the left. The shift rows operation increases the properties of linearity influences.

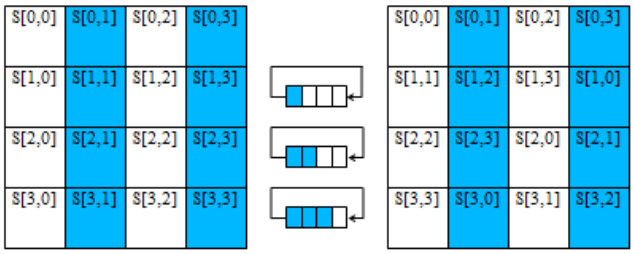

Figure 6*: Basic transformations of the shift rows function.*

Unlike the shift rows operation being the first procedure of the encryption process, the inverse shift rows operation is first of the decryption process. Similar to the shift rows operation in the encrypt process.

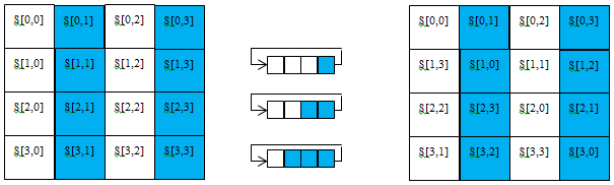

Figure 7*: basis transformations of the inverse shift row operation*

Figure 7, demonstrates the basis transformations of the inverse shift row operation. The first row is not shifted, the second row is shifted once, the third row is shifted twice and the fourth row is shifted three times. However, the shifts in the decrypt process will be done to the right as.

The Mixed Columns component is responsible for operating on each column of the state. The values of the *state* are polynomials of the *GF(2^8)*. Each column of the *state* undergoes a transformation by multiplying modulo $x^4 + 1$ against an established polynomial resulting in the matrix shown in Figure 8. Each column is multiplied against the 4x4 matrix. This operation is processed until the ***Nr-1*** round.

$$\begin{bmatrix} S'[0,c] \\ S'[1,c] \\ S'[2,c] \\ S'[3,c] \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S'[0,c] \\ S'[1,c] \\ S'[2,c] \\ S'[3,c] \end{bmatrix}$$

Figure 8*: Transformation Matrix*

With the inverse mixed columns used in the decryption process, the same logic is used in this operation as it was used in the encryption operation. Figure 9, represents the 4x4 matrix used against the four byte input values of each column. The Add Round Key was responsible for adding the Round Key that was obtained from the key schedule to the *state* utilizing a bitwise XOR operation.

$$\begin{bmatrix} S'[0,c] \\ S'[1,c] \\ S'[2,c] \\ S'[3,c] \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} S'[0,c] \\ S'[1,c] \\ S'[2,c] \\ S'[3,c] \end{bmatrix}$$

Figure 9*: 4x4 matrix used against the four byte input values of each column*

3. AES ATTACKS

The only known attacks on the AES algorithm are side-channel attacks and attacks on weaknesses found in implementation or key management. The AES algorithm must be implemented in the correct strategic way as required by the AES standards. The first key-recovery attacks on full AES were due to Bogdanov et al [2]. The attack is a biclique attack and is faster than brute force by a factor of about four. It requires 2126.2 operations to recover an AES-128 key. Bogdanov et al concludes the following results.

- The first key recovery attack on the full AES-128 with computational complexity $2^{126.1}$.
- The first key recovery attack on the full AES-192 with computational complexity $2^{189.7}$.
- The first key recovery attack on the full AES-256 with computational complexity $2^{254.4}$.
- Attacks with lower complexity on the reduced-round versions of AES not considered before, including an attack on 8-round AES-128 with complexity $2^{124.9}$.
- Preimage attacks on compression functions based on the full AES versions

The National Security Agency (NSA) expects that 256 bit AES keys may be cracked by 2018.

4. STEGANOGRAPHY

Steganography conceals the secret message in plain sight through the use of a cover object. Steganography can be used for concealing and protecting the objects. Concealing is being able to hide the secret message in a cover object without an adversary being aware of any information embedded. Protecting, however, is used in situations where media data has to be protected. Many people use this technique of steganography, such as digital watermarking, in order to protect the original works of authorship to author.

Although steganography has various techniques, the experiments of this paper utilize the least significant bit

insertion method for each trial. The two common used methods in steganography are the least significant bit insertion method and the Discrete Cosine Transformation (DCT) method. The least significant bit insertion method is the least complex of the two, being capable of accessing the RGB of each pixel (picture element). The objective of this technique is to swap the least significant bit of each color in every pixel with the bit of the secret data. Because of its low level of complexity the LSB insertion method does not provide efficient robustness. This is vulnerable to transformations such as modifications. In addition to file compressions, a jpeg file format could not withstand this technique, losing information for the extracting process. For the experiments of this paper, a bmp file format was utilized. However, complex mathematically algorithms can be developed to provide scattered positions for the secret data.

Steganography is profoundly a unique process. Unlike other tests (based on speed), the steganography demonstration in this paper is measured by three characteristics. These trials had to satisfy these conditions. First, after embedding the secret data within the cover object is to protect the integrity of the Stego object. Once the Stego object is sent through the data communication medium, a technique is needed to protect the embedded data at the point of retrieval. Next, the stego object must be indistinguishable from the cover object. Once the secret information has been embedded within the cover object, the Stego object should be completely identical to the cover object, displaying no signs of variations. Finally, the extracting process should be accurate. After the receiver has received the stego object and performed the extracting operation, the message should be returned in the original form (sent from source). These characteristics will determine the success of the stenographic operation.

5. RELATED WORK

Understanding the basics of cryptography, finite fields, and steganography were important to the development of the AES algorithm and steganography. The reference [3 - 6] provides the basics. Manoj et al. [7] performed AES based steganography. Manoj utilized steganography with biometrics. The work uses skin tone for the embedding process. The secret data was embedded in one of the high-frequency sub-band of DWT. Data hiding was achieved by cropping an image interactively.

The authors in [8] discussed the secure steganography approach using AES. They used AES-128 and utilized least two significant bits for transformation. The current model uses 256 bits and S-Box for steganography which differs from Ramaiya [8]. The image analysis is characteristic in both cases.

Recently, Arjun et al. [9] discussed steganography based AES algorithm. The authors presented limited literature on AES and usage of LSB function was not seen. The authors used bit plane complexity for steganography technique. This method divided the image into bit planes before using AES algorithm.

6. IMPLEMENTATION AND RESULTS

In our implementation experiments, a simple 55 byte text file was utilized to demonstrate the operations of enhancing the security of a secret message using cryptography and steganography, which are implemented using the step-down procedures below.

- Because the AES-256 is the desired cryptographic function, the first step is to create a 32-bit key and encrypt the data using the AES-256 algorithm.
- Next, the cover object is loaded for the placement of the encrypted message.
- Once the image is uploaded, the intensity of each pixel is then entered into the access domain. During this process, the Least Significant Bit (LSB) of each required pixel of the Red, Green, Blue color is available for the embedding alteration.
- A stego key is then created in order to conceal the encrypted message. The stego key will be used to grant permission to proceed with extracting process.
- The secret message is then converted into its binary form in so that each bit can be embedded into the Least Significant Bit binary value of each color.
- The message length of the encrypted message is obtained. This will allow the receiver to recover the positions of where the secret message starts and ends in the extracting process.
- Each bit of the encrypted file is then placed at the Least Significant Bit value of each color in the pixels altering the color value slightly.
- The stego object is then sent to the receiver and once received; the receiver will use the stego key in order to perform the extracting operation of the encrypted message.
- After the encrypted message is extracted it is then decrypted using the inverse operation of the AES algorithm.

By using the above procedures, **Figure 10** below displays the results of the experiment.

Original Message

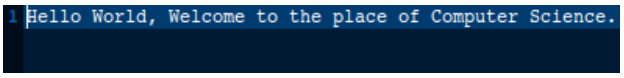

Figure 10 (a)

Encrypted Message

Figure 10 (b)

Medial File

Cover Object

Stego Object

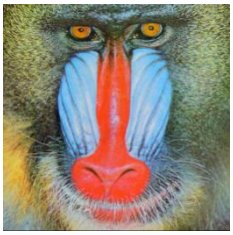

Figure 10 (c)

Decrypted Message

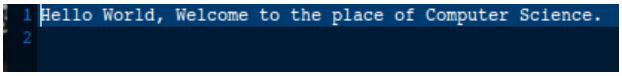

Figure 10 (d)

- **Figure 10 (a)** displays the original message (*Hello World, Welcome to the place of Computer Science*.) that was used for the encrypting process using the AES-256 cryptographic function. Since the AES-256 was utilized a key of 32 bytes was created. Note: it is never advised to have the key set in the program but for demonstration purposes it was created inside the program. Once finished, the encrypted message was generated which is found in **Figure 10 (b)**.
- **Figure 10 (c)** displays two identical images. However, these two images are different. The image on the left is the cover object or the media file that was loaded for the placement of the encrypted message. In this testing, the cover object was a bitmap image file of about 786.5KB in size. The intensity of each pixel was entered into the access domain for the embedding process.
- The stego key was created in order to allow the program to process the extracting process. After the stego key was created, the encrypted message was the converted to its binary form in order to be embedded into each color inside of each pixel.
- The Least Significant Bit (LSB) of each required pixel of the Red, Green, Blue color was available for the embedding alteration. Here a Stego key was created in order to conceal the secret message.
- The message length of the secret file was obtained for the positions of where the secret message starts and ends for the steganography decoding process.
- Each bit of the encrypted message was then placed at the Least Significant Bit of the color value of each pixel, altering the color value slightly.
- After the embedding process was completed the resulting image was generated with the encrypted message embedded into it which can be found on the right of **Figure 10 (c)**. In comparison, the stego object must remain unchanged to the human eye. Capacity of each image was compared to test for any changes in the size. When this procedure was tested there was no different in size, resulting in the image appearing unchanged to the human eye.
- After the stego object reached the receiver's end, the stego key was applied in order to perform the decode process to extract the encrypted message.
- The encrypted message was then decrypted using the inverse operation of the AES algorithm giving the results of **Figure 10 (d).** As one can see the decrypted message derived back to the original message resulting in the procedure being a success.

7. CONCLUSION

We conducted the series of experiments to make sure our algorithm works as desired. There is more to explore about the field of Cryptography and Steganography. Because these are the most common foundations of the two components more work will be done to improve effectiveness and efficiency.

Future work will include implementing public key cryptography to enhance the security of the data in transit. Also, experiments and techniques will be developed for processing a steganography program at the cloud level for the improvement of cloud security.

REFERENCES

[1] J. Daemen and V. Rijmen, "AES Proposal: Rijndael, AES Algorithm Submission", September 3, 1999 (http://www.nist.gov/CryptoToolkit)

[2] A. Bogdanov; D. Khovratovich and C. Rechberger, "Biclique Cryptanalysis of the Full AES", Volume 7073 of the series Lecture Notes in Computer Science, 2011, pp 344-371.

[3] Paar, Christof, and Jan Pelzl. Understanding Cryptography. Springer, 2010.

[4] Shashikala Channalli, Ajay Jadhav. "Steganography An Art of Hiding Data". International Journal on Computer Science and Engineering, Vol.1 (3), 2009, 137-141.

[5] J. Buchmann., "5 DES", Introduction to cryptography, Springer, 2001, pp. 119-120.

[6] D. Canright., "A very Compact S-Box for AES", Springer, Volume 3659, Lecture Notes in Computer Science, 2005, pp 441-455.

[7] M. Gowtham, et al., "AES based Steganography", IJAIEM, vol.2, issue 1, 2013, pp 382-389.

[8] M. Ramaiya, et al., "Secured Steganography Approach Using AES", IJCSEITR, vol 3, issue 3, 2013, pp 185-192.

[9] A. Kumthe, et al., "Steganography based on AES Algorithm and BPCS Technique for a Securing Image", IJARCSSE, vol 6, issue 3, 2016, pp 116-119.