INFOKUM
Volume 13, Number 05, 2025, DOI 10.58471/infokum.v13i05
ESSN 2722-4635 (Online)
https://infor.seaninstitute.org/index.php/infokum

Steganography Test for Exif Metadata on JPEG Files With AES-256 Encryption for Secret Message Security

Yasir Hasan
Program Studi Informatika, STMIK Mulia Darma, Labuhanbatu, Sumatera Utara, Indonesia

Article Info
Keywords:
Steganography
AES-256
Base64
EXIF Metadata
JPEG Files

ABSTRACT
One of the main challenges is hiding secret messages in media such as JPEG images. However, embedding messages directly into EXIF metadata can pose detection risks and vulnerability to forensic analysis attacks, and this technique faces challenges in security, resistance to metadata manipulation, and message extraction accuracy. The method used in this study involves embedding a message encrypted using AES-256 ECB mode and Base64 into the EXIF metadata of JPEG files, so that only those with the decryption key can access the message content. This system is designed to be compatible with standard image processing software without changing the main structure of the JPEG file, making it difficult to detect by conventional metadata analysis techniques. Test results show that this method is able to embed secret messages with a high level of security without changing the visual quality of the image. AES-256 encryption encoded in Base64 is proven to be effective in maintaining the confidentiality of messages, so that only users with the correct decryption key can access them. Thus, the combination of EXIF metadata steganography and AES-256 encryption in Base64 provides an effective solution for securing secret messages in JPEG files, improving data protection against the threat of information theft and manipulation.

Corresponding Author:
Yasir Hasan
Program Studi Informatika, STMIK Mulia Darma, Labuhanbatu, Sumatera Utara, Indonesia
yasirhasan.kom@gmail.com

INTRODUCTION

Steganography is scientifically developed as a technique for inserting messages into images, audio, video, or other files. [1]. This technique differs from cryptography in that it not only secures the message content but also conceals its existence. In academia and cybersecurity, steganography is used to protect data from unauthorized eavesdropping or modification.[2], [3]. However, beyond scientific applications, steganography is not only useful for security, but can also be a fun tool in various creative and entertainment aspects. In digital art, artists can embed hidden messages in images or music that can only be discovered using specific methods. In the gaming world, steganography is often used to hide secret clues or puzzles, making the game more interesting. In hacking competitions such as Capture the Flag (CTF), steganography is often used to hide flags or important clues, challenging participants to find the hidden information. [4], [5]. With these creative applications, steganography is a fun way to add an element of mystery and challenge to the digital world.

Exchangeable Image File Format (EXIF) metadata is a part of an image file that stores information about the photo taken, such as the date and time, camera model, GPS location, and other technical parameters. [6], [7], [8]EXIF data can be easily edited or modified using various software, allowing for the insertion of additional information into the metadata without altering the visual appearance of the image. This makes EXIF metadata a potential medium for hiding secret messages with steganography compared to JFIF and XMP. [3], [9]. In addition to flexibility in modification, EXIF metadata is also retained by most image processing software, making it an attractive option in research on steganography and data security.

The Advanced Encryption Standard (AES) algorithm, which uses a 256-bit key, is one of the strongest encryption methods used in various data security applications. AES-256 has been adopted as a standard and is widely used in the industry to protect sensitive data. [10], [11]. It uses a complex substitution-permutation principle, making it extremely difficult to crack without knowing the correct encryption key. The encryption process is performed using Electronic Codebook (ECB) mode. [12]. This mode was chosen because of its simple structure and is suitable for short text data that will be inserted into metadata. [13]. After encryption, the resulting ciphertext is converted back to Base64 format to be compatible with the text structure in EXIF metadata and so that it does not contain binary characters that can corrupt the file format. In this study, AES-256 is used to encrypt the message before it is embedded into the EXIF metadata, so that even if the hidden message is discovered, the message content remains protected from unauthorized access. [14], [15].

To ensure that encrypted messages can be easily recognized and extracted, it is necessary to create special markers in the EXIF metadata. [7], [16]. These markers serve as indicators that the metadata contains encrypted information that requires further processing. Without these markers, the extraction process would be difficult because EXIF metadata typically contains a lot of information irrelevant to steganography. Inserting these markers must be done carefully to avoid attracting attention or causing noticeable changes to the metadata that could arouse suspicion. With systematically designed markers, the information insertion and extraction process become more efficient and structured.

Testing in this study was conducted to assess the effectiveness and security of the proposed method. The trials included successful message insertion and extraction, robustness to metadata modification, and the method's ability to withstand digital forensic analysis. Furthermore, this study examined the impact of encryption on file size and compatibility with various image processing software. By conducting comprehensive testing, this research is expected to contribute to the development of more secure steganography techniques that can be applied to various confidential communication needs in the future.

METHOD

The system design includes the message embedding and extraction process, as well as the placement of encrypted messages in the EXIF metadata field. When reading the initial input file, the AES-256 encrypted secret message marker is detected in the file. If the marker is found, the file is extracted; if not, the encrypted message is embedded.

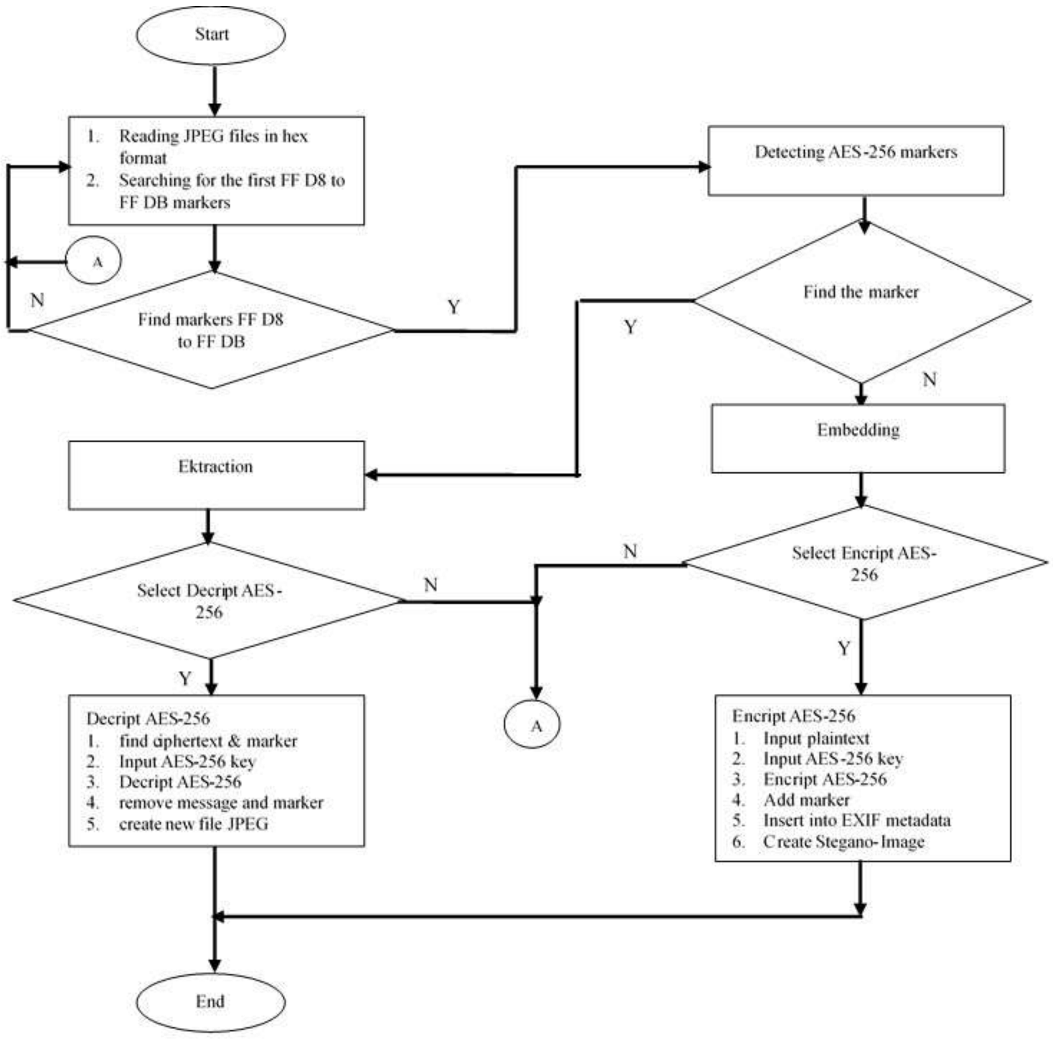


Figure 1. Steganography flow and secret message security

Secret Message Embedding and Encryption

1. Input
   JPEG File: A digital image containing EXIF metadata and used as the embedding medium.
   Secret Message: The text to be hidden (Plaintext). Encryption Key: A 256-bit key for the AES-256 encryption process.
2. Process
   Message Encryption Algorithm: AES-256 (Advanced Encryption Standard) ECB mode and Base64 encoding. Marker Creation: Markers are used to mark the beginning and end of the encrypted message in the metadata. Embedding Process: Modifying EXIF Metadata: The EXIF field is selected and the encrypted message plus marker is inserted.
3. Output
   Save Image File: The JPEG file with the modified metadata is saved as a new output (stego-image).

Secret Message Extraction and Decryption

1. Input
   JPEG: A digital image containing EXIF metadata and used as an embedding medium. Secret Message: The encrypted text (Ciphertext). Decryption Key: A 256-bit key for AES-256 decryption.
2. Process
   Extraction Process: Separate the marker from the encrypted message, Base64 Decoding and reading the ECB block, Decrypting the Message Algorithm: AES-256, Removing the marker and message.
3. Output
   Save Image File: A JPEG file with metadata, but without the secret message and marker, is saved as a new output (new-image).

Evaluation and Analysis of Results

Evaluation of image file compression results is carried out by measuring several key parameters that reflect the effectiveness of the applied method. Bit Error Rate (BER) testing indicates the bit error rate. Payload capacity refers to the maximum amount of data that can be inserted into a metadata field. Robustness is evaluated based on the ability of hidden data to remain intact even if the image file undergoes compression or light manipulation. Meanwhile, imperceptibility is assessed by the extent to which the presence of a hidden message does not disrupt the visual appearance of the image and is undetectable to the naked eye or by common software, ensuring the security of the inserted message. Finally, against brute-force attacks, calculating the probability of the message being practically cracked with current computing power.

RESULTS AND ANALYSIS

Evaluation

Testing was conducted to evaluate the effectiveness of the EXIF metadata steganography system with AES-256 ECB encryption and Base64 encoding in terms of message extraction accuracy, storage capacity, visual quality, and resistance to metadata manipulation. Testing was conducted using several common scenarios reflecting real-world situations, such as file recompression, metadata removal, and format conversion on three image files. The plaintext and key messages were also varied.

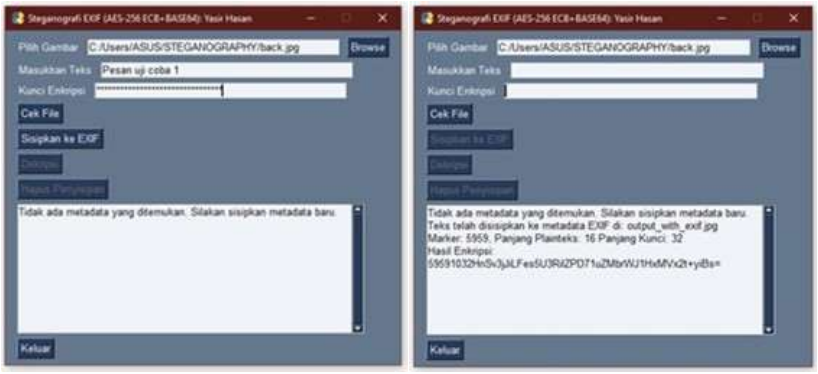
From the test results for all image files, only the back.jpg file, converted to output_with_EXIF_1.jpg, and the STMIK MD.jpg file, converted to output_with_EXIF_2.jpg, were successfully embedded, leaving the image intact. The embedding of the hp.png file, converted to output_with_EXIF_3.jpg, was still successful, but resulted in file changes and corruption. This is because the application's embedding concept is not specific to PNG image files, but only JPG or JPEG files. The JPEG file structure consists of several key components that determine how the image is interpreted by the software. [24]. Each part in a JPEG file has a specific role [22], [23].

Table 1. Input image file, Plaintext, Key and Ciphertext

File gambar	Plaintext	Key	Message with marker
back.jpg (10.7 KB)	Pesan uji coba 1	STMIK Mulia Darma Labuhan Batu 1	Ciphertext1: 59591032HnSv3jJiLFes5U3RiIZPD1uZMbrWJ1HxMVx2t+yiBs= (Size = 52 Byte)
STMIK MD.JPG (127 KB)	3n4m B3l45	STMIK Mulia Darma Labuhan Batu 1	Ciphertext2: 59591032q321/BQcH2Ltwcc+WGx1Kw== (Size = 32 Byte)
bongkar hp.PNG (1.02 MB)	Percayalah hidup tidak seindah yang kau bayangkan jika tanpa agama	STMIK Mulia Darma Labuhan Batu 2	Ciphertext3: 59591032Z2t9YZVTYf6Jp4oU0WI+LIkc4oQpthUkQ256I1Ft+xF04JuuzNnCTafGAgYquliq3Puo4YNmq7Xm912A4imY/03vGfLakv68QvTLbvZpuz8= (Size = 96 Byte)

Running Python Program Code

Implementation of EXIF metadata steganography application on JPEG image files using AES-256 encryption in ECB mode and PySimpleGUI based graphical user interface. This program allows users to insert secret messages into image files through metadata, with an encryption process based on the AES algorithm and Base64 encoding. The message to be inserted is first processed with padding to fit into a 16 Byte AES block, then encrypted and given a special marker in the form of the string "5959" to facilitate metadata search. All of these processes are controlled through a user-friendly interface, including image input, text, encryption key, and action buttons such as Check File, Insert into EXIF, Decrypt, and Delete Insertion.



Pilih Gambar: C:/Users/ASUS/STEGANOGRAPHY/back.jpg
Masukkan Teks: Pesan uji coba 1
Kunci Enkripsi: ********************************

```
Tidak ada metadata yang ditemukan. Silakan sisipkan metadata baru.
```

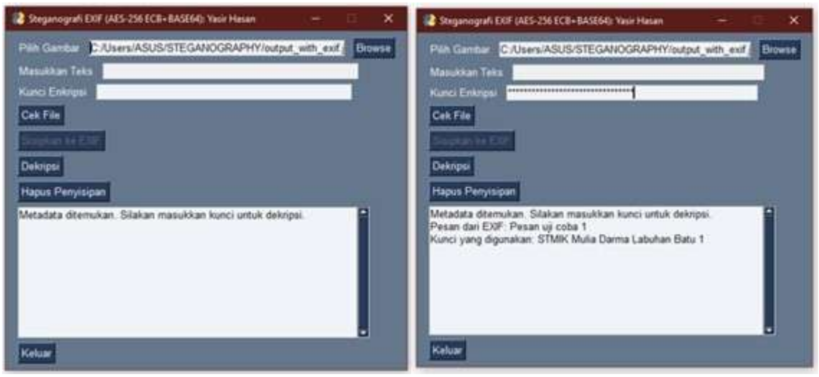
Pilih Gambar: C:/Users/ASUS/STEGANOGRAPHY/back.jpg

```
Tidak ada metadata yang ditemukan. Silakan sisipkan metadata baru.
Teks telah disisipkan ke metadata EXIF di: output_with_exif.jpg
Marker: 5959, Panjang Plainteks: 16 Panjang Kunci: 32
Hasil Enkripsi:
59591032HnSv3jJiLFes5U3RiIZPD71uZMbrWJ1HxMVx2t+yiBs=
```

Figure 2. (left) File selection, marker check, and message and key input, (right) message embedding process

This program begins by selecting an image file and checking for a marker. If the marker is not found, the input message and key are provided. The next step is to embed the ciphertext into the selected image file.

The program also provides features to detect whether encrypted metadata already exists, decrypt the message from the metadata using the same key, and delete the embedded message. The output file is saved with a unique name to prevent overwriting the original file. The extraction process is the same as the embedding process in the initial section, except that if the marker is found, the next step is to enter the decryption key. Alternatively, if you do not want to decrypt the message, you can simply delete it from the file without decrypting it.



```
Metadata ditemukan. Silakan masukkan kunci untuk dekripsi.
```

```
Metadata ditemukan. Silakan masukkan kunci untuk dekripsi.
Pesan dari EXIF: Pesan uji coba 1
Kunci yang digunakan: STMIK Mulia Darma Labuhan Batu 1
```

Figure 3. (left) File selection and marker check, (right) key input and message extraction process

Bit Error Rate (BER) Testing

Bit Error Rate (BER) is used to measure the message error rate of all decrypted Ciphertexts compared to the original message. [17]. BER is calculated as the ratio of the number of incorrect bits to the total number of bits in the file.

From the BER results, all ciphertexts have a value of 0, meaning there is no error rate. However, only the file [18].

Table 2. Bit Error Rate (BER) Testing

No	Input File	Output File	Decryption Result	BER (%)	Description
1	back.jpg (10.7 KB)	output_with_EXIF_1.jpg (10.7 KB)	Same	0.00	- Success -File intact

2	STMIK MD.jpg (127 KB)	output_with_EXIF_2.jpg (127 KB)	Same	0.00	- Success -File intact
3	bongkar hp.PNG (1.02 MB)	output_with_EXIF_3.jpg (1.02 MB)	Same	0.00	- Success -File corrupted

Payload Capacity Testing

Payload capacity indicates the maximum amount of data that can be inserted [19]. Testing is performed by inserting messages of varying sizes and measuring the maximum metadata limit that can still be accommodated without failure. EXIF metadata is stored in JPEG segments called APP1 (Application Marker 1) or other. The JPEG format limits the size of each marker segment to a maximum of 64 KB: 65533 bytes, of which the first 2 bytes are used for the segment length, i.e., 65535 - 2 = 65533.

Tabel 3. Payload capacity Testing

EXIF Field	Maximum Capacity	Maximum Length (Bytes)	Characters Message	Description
User Comment	Height	64,000 Bytes (with padding)	60,000–63,000 characters	- EXIF field for user comments - suitable for large payloads or ciphertext
Image Description	Medium	2,048 Bytes (software dependent)	2.000 characters	- for image description - often truncated by editing software or cameras

Imperceptibility Testing

Imperceptibility testing refers to how visually undetectable changes made to a JPEG file are [20], [21]. Because the embedding is performed only on metadata, there are no visual changes to the image. The embedded file can still be displayed in standard image viewers without issue. However, the output_with_EXIF_3.jpg file experiences physical changes; information that should be displayed in digital image format is corrupted, altered, and unavailable. This is because the input file is in PNG format. The other output files, however, remain unchanged, with only the addition of secret message data. Based on the results of this test, the input image file must be in JPG or JPEG format. PNG files are not acceptable for use as input image files for this steganography.

Table 4. Imperceptibility Testing

No	Output File	Size File	Visual Appearance	Compatibility
1	output_with_EXIF_1.jpg	Increased by 52 Bytes	- Same as input file - No change	- Can be opened with image application - Editable
2	SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER STMIK MULIA DARMA output_with_EXIF_2.jpg	Increased by 32 Byte	- Same as input file - No change	- Can be opened with image application - Editable
3	output_with_EXIF_3.jpg	Increased by 96 Byte	- Not the same as input file - Change	- Can be opened with image application - No image information

Robustness Testing

Robustness testing measures the system's resilience to tampering such as metadata deletion, file compression, or format conversion on output files[20], [21], [22]. Results indicate that the system is quite robust to light editing, but is known to be vulnerable to complete metadata deletion of output files. Furthermore, it is vulnerable to format conversions that do not preserve EXIF.

Table 5. Robustness Testing

Scenario	output_with_EXIF_1.jpg	output_with_EXIF_2.jpg	output_with_EXIF_3.jpg	Description
Metadata intact	High	High	High	Message decryption successful
Metadata partially removed	Medium	Medium	Medium	Can still be decrypted
Metadata completely removed	Low	Low	Low	Message cannot be found
JPEG	Medium	Medium	Medium	corrupted message,

recompressed				can be partially decrypted
JPEG → PNG → JPEG	Low	Low	Low	Metadata completely lost

Security and Reliability Evaluation of the Method

A security and reliability analysis was conducted to assess the extent to which the EXIF metadata steganography method combined with AES-256 encryption is able to protect secret messages and resist tampering or manipulation. The two main aspects analyzed are the cryptographic security of the algorithm used and the technical reliability of the steganography method under various file conditions. Steganographic security depends on the imperceptibility of changes to the media used (imperceptibility) and resistance to changes or manipulation (robustness) [21]. In some cases, steganography is combined with encryption to enhance security, so that even if the hidden message is detected, its contents remain unreadable without the correct encryption key [18].

AES-256 Security Evaluation

AES-256 is a cryptographic algorithm with a 256-bit key length, making it highly resistant to brute-force attacks. In this implementation, the use of ECB mode does pose a weakness in identical block patterns, but in the context of text metadata that does not have repeating patterns, such as digital images, this risk can be considered minimal.

The use of the "5959" marker does not compromise security because it does not contain key information or the encryption structure. The marker serves only as an indicator of the presence of ciphertext, without revealing the contents or details of the cryptographic system. Encryption is performed before embedding into the metadata, so even if the metadata is read by a third party, the visible data remains in the form of meaningless ciphertext without the key.

Brute Force Attack Analysis

A brute force analysis of ciphertext encrypted using AES-256 in ECB mode shows that, even if the ciphertext is known in Base64 format, brute force is still practically impossible. The complexity of the ciphertexts of the two algorithms, AES-256 and Base64, already guarantees the message's resistance to brute force. AES-256 has a key space of 2^{256} combinations, meaning it would take approximately 10^{49} years to try all possible keys, even with the fastest supercomputer capable of 10^{20} attempts per second [23]. The length of the ciphertext, whether 16 bytes, 32 bytes, or 66 bytes, does not affect the difficulty of brute force because complexity is determined by key length, not message length. Therefore, without knowledge of the key or the characteristics of the plaintext, all ciphertexts are theoretically and practically secure against brute force.

Table 6. Brute force calculation of messages without markers

Ciphertext	Size Base64	Ciphertext Size	Brute force Time AES-256 + Base64
Ciphertext1	44 characters	32 Byte	~10^{49} years
Ciphertext2	24 characters	16 Byte	~10^{49} years
Ciphertext3	88 characters	66 Byte	~10^{49} years

Metadata Embedding Reliability Evaluation

Based on the test results, the reliability of this method is also demonstrated by:

1. Imperceptibility
   High imperceptibility, where the embedded JPEG files show no visual changes and can still be opened normally. Furthermore, the presence of markers allows the system to automatically check for the presence of hidden data, making the extraction process more targeted and efficient.
2. Payload Capacity
   There are limitations to the payload capacity, due to the very limited maximum length of EXIF metadata. For JPEG files with small metadata space or that are already fully used by the camera, embedding long messages may fail. Therefore, this system is more suitable for embedding short and sensitive messages, rather than large documents.
3. Robustness
   Messages can still be extracted and decrypted correctly if the EXIF metadata remains intact or undergoes only minor changes. However, this method is not robust to extreme compression or file format conversions that automatically remove metadata, such as those in social media uploads that can change or convert JPEGs to PNG or other formats.

The system developed in this research aims to hide secret messages into JPEG image files by inserting them into EXIF metadata, then encrypting the message using the Advanced Encryption Standard (AES-256) algorithm in Electronic Codebook (ECB) mode, where each Plaintext block is processed independently to produce a Ciphertext with a fixed size. The selection of ECB here is to show the systematics of AES in real terms which is repeated according to the length of the same 16 Byte Plaintext block. Finally, Base64 is applied for Ciphertext encoding.

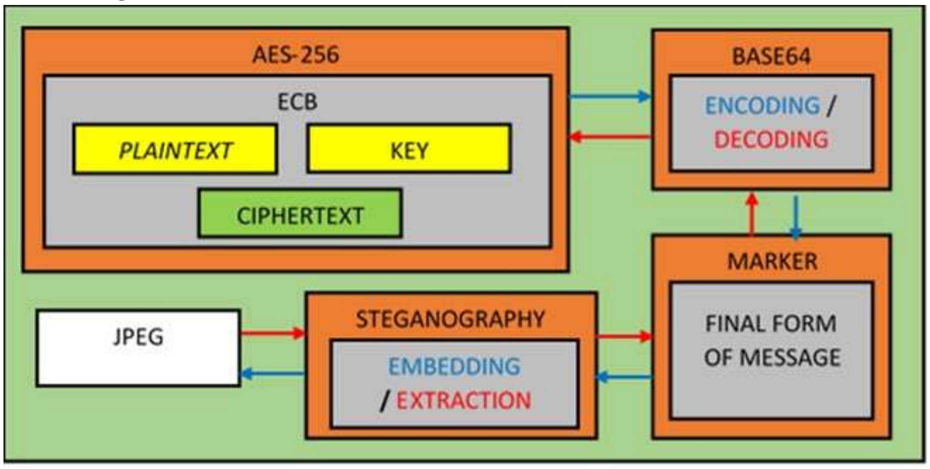


Figure 4. Message encoding and steganography scheme in JPEG files

AES-256 ECB Mode Encryption and Base64 Encoding (Embedding)

This study uses the AES-256 ECB mode algorithm to encrypt secret messages before embedding them into the EXIF metadata of JPEG files. In addition, AES-256 also creates an AddRoundKey (0), 13 rounds (SubBytes, ShiftRows, MixColumns, AddRoundKey), and a final round (SubBytes, ShiftRows, AddRoundKey)[27]. During the encryption process, AES-256 also uses S-BOX and key schedule generation, similar to the Rijndael algorithm. The following is a test of the AES-256 encryption process with plaintext, key, and process:

Plaintext 16 Bytes: "Test message 1"
Converted to Hex 506573616E20756A6920636F62612031
Key 32 Bytes: "STMIK Mulia Darma Labuhan Batu 1"
Converted to Hex 53544D494B204D756C6961204461726D61204C61627568616E20426174752031

In ECB mode, it is necessary to change the AES Plaintext in the form of a block cipher so that AES only works on data in a fixed size per block (AES → 128-bit = 16 Bytes). If the Plaintext is not a multiple of 16 Bytes, then padding must be added and if the Plaintext is exactly 16 Bytes, then apply Public-Key Cryptography Standards #7 (PKCS#7) for the standard padding scheme used in block cryptography which still adds 1 full block of padding (16 Bytes 10 hexadecimal 16 times) to prevent ambiguity during decryption later. Thus, the plaintext becomes two blocks:

Plaintext: block 1 16 bytes + block 2 16 bytes
Block 1 "Test message 1"
Block 2 "\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10\x10"
Block 1 Hex 506573616E20756A6920636F62612031
Block 2 Hex 101010101010 10101010101010101010

The next step is encryption with 14 rounds. First, AddRoundKey 0 is obtained. AddRoundKey 0 is derived from the master key with a key schedule. For this, we obtain each AddRoundKey 0:
AddRoundKey 0 block 1 = "43 44 5D 59 5B 30 5D 65 7C 79 71 30 54 71 62 7D"
AddRoundKey 0 block 2 = "03 31 3E 28 25 00 38 1F 05 49 02 4F 26 00 52 5C"

Table 7. 14 Round AES Block 1 ECB

Round 1	SubBytes	7	C	B	3	3	6	0	C	6	3	7	8	F	6	0	4
	ShiftRows	7	6	7	4	3	3	0	3	6	6	B	C	F	C	0	8
	MixColumn	6	6	2	0	0	7	5	1	0	A	2	F	2	E	A	D
	AddRound	0	4	6	6	6	0	3	7	6	8	6	9	5	9	8	E
Round 2	SubBytes	7	2	4	5	4	3	0	8	A	C	9	9	5	B	A	E
	ShiftRows	7	3	9	E	4	C	A	5	A	B	4	8	5	2	0	9
	MixColumn	C	4	5	E	2	6	2	1	5	8	1	1	4	9	D	E
	AddRound	0	A	D	3	A	A	E	B	B	2	B	9	E	5	0	0
..	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Round 13	SubBytes	3	2	9	4	9	2	1	2	0	D	B	5	3	2	9	D
	ShiftRows	3	2	B	D	9	D	9	4	0	2	9	2	3	2	1	5
	MixColumn	6	6	1	6	9	D	A	7	D	D	6	F	5	1	C	D
	AddRound	F	A	7	D	5	C	B	0	4	7	0	4	6	4	6	E
Round	SubBytes	4	0	4	5	F	B	8	0	6	4	6	E	A	0	3	C
	ShiftRows	4	B	6	C	F	4	3	5	6	0	4	0	A	0	8	E
	AddRound	1	7	A	D	3	6	2	5	A	E	4	D	8	8	4	0

Next below is all the same process for block 2 after AddRoundKey 0.

Table 8. 14 Round AES Block 2 ECB

Round 1	SubBytes	1	1	4	C	3	0	4	4	1	B	A	0	2	A	A	F
	ShiftRows	1	0	A	F	3	B	A	C	1	A	4	4	2	1	4	0
	MixColum	6	1	5	6	D	6	8	D	D	D	F	4	2	C	A	3
	AddRoun	0	3	1	0	B	1	E	B	B	F	B	2	5	B	8	0
Round 2	SubBytes	6	C	5	D	E	5	2	E	C	B	A	3	D	6	7	A
	ShiftRows	6	5	A	A	E	B	7	D	C	6	5	E	D	C	2	3
	MixColum	3	9	9	0	A	C	D	4	8	1	3	B	F	0	1	E
	AddRoun	F	7	1	D	2	0	1	E	6	B	9	3	5	C	C	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
Round 13	SubBytes	0	A	9	7	2	E	B	8	D	3	2	F	7	7	6	B
	ShiftRows	0	E	2	B	2	3	6	7	D	7	9	8	7	A	B	F
	MixColum	A	1	6	A	0	9	4	D	3	1	0	9	4	1	B	7
	AddRoun	3	D	0	1	C	8	5	A	A	B	6	2	7	4	1	4
Round	SubBytes	E	3	7	4	2	A	4	0	0	7	A	F	F	2	8	D
	ShiftRows	E	A	A	D	2	7	8	4	0	2	7	0	F	3	4	F
	AddRoun	B	6	6	C	E	5	9	4	C	C	7	D	D	B	8	1

After the encryption process, the AddRoundKey 14 result for each table is converted into Ciphertext block 1 plus Ciphertext block 2.

Ciphertext block 1 = "1E74AFDE32622C57ACE54DD188864F0F"

Ciphertext block 2 = "BD6E64C6EB589D47C4C571DADFB2881B"

The ciphertext is encoded in Base64 format to maintain compatibility with the text-based EXIF metadata format. This process converts each three bytes of binary data into four ASCII characters consisting of only 64 specific symbols.[28]. The following is the result of base64 encoding and adding markers to the ciphertext.

ECB ciphertext:
"1E74AFDE32622C57ACE54DD188864F0FBD6E64C6EB589D47C4C571DADFB288 1B"

Encoding Base64:
"HnSv3jJiLFes5U3RiIZPD71uZMbrWJ1HxMVx2t+yiBs="

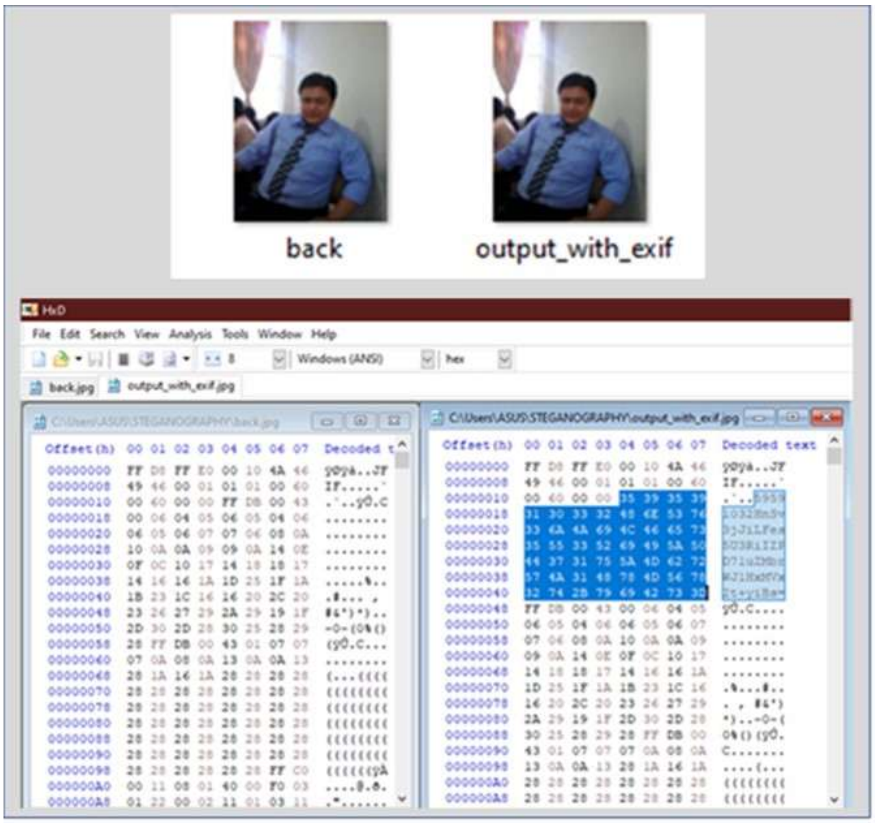
Insertion into EXIF Metadata

Encryption results are inserted at the end of the EXIF metadata, specifically in the field after EXIF and before the "FF DB" header. This field is chosen because it is used to add additional text information and can accommodate sufficient data length for short to medium ciphertext. Before inserting the Base64 ciphertext into the metadata, a special marker is added, consisting of the hexadecimal string "5959" with the value "35 39 35 39" at the beginning of the data. Also, add the plaintext length and key length after the "5959" marker. The plaintext length is 16 bytes for each block, and the key length is 32 bytes. This marker serves as an identifier that the data in the metadata is AES-256 encrypted and facilitates detection and extraction during decryption. The following is the result of adding markers, plaintext length, and key length:

ECB Ciphertext:
1E74AFDE32622C57ACE54DD188864F0FBD6E64C6EB589D47C4C571DADFB2881 B

Base64 Encoding:
HnSv3jJiLFes5U3RiIZPD71uZMbrWJ1HxMVx2t+yiBs=
Ciphertext with markers:
59591032HnSv3jJiLFes5U3RiIZPD71uZMbrWJ1HxMVx2t+yiBs=

This ciphertext is inserted into the back.jpg image file right at the end of the EXIF section or before the image file content data. The following uses the HxD application to show evidence of Ciphertext data that has a marker inserted in the back.jpg image below, which is visible in blue as a Ciphertext selection mark that has a marker and at the top of the image there is an input image back.jpg and an image of the results of inserting Ciphertext with the name output_with_EXIF.



```
C:\Users\ASUS\STEGANOGRAPHY\back.jpg

Offset(h)  00 01 02 03 04 05 06 07  Decoded text
00000000   FF D8 FF E0 00 10 4A 46  ÿØÿà..JF
00000008   49 46 00 01 01 01 00 60  IF.....`
00000010   00 60 00 00 FF DB 00 43  .`..ÿÛ.C
00000018   00 06 04 05 06 05 04 06  ........
00000020   06 05 06 07 07 06 08 0A  ........
00000028   10 0A 0A 09 09 0A 14 0E  ........
00000030   0F 0C 10 17 14 18 18 17  ........
00000038   14 16 16 1A 1D 25 1F 1A  .....%..
00000040   1B 23 1C 16 16 20 2C 20  .#... , 
00000048   23 26 27 29 2A 29 19 1F  #&')*)..
00000050   2D 30 2D 28 30 25 28 29  -0-(0%()
00000058   28 FF DB 00 43 01 07 07  (ÿÛ.C...
00000060   07 0A 08 0A 13 0A 0A 13  ........
00000068   28 1A 16 1A 28 28 28 28  (...((((
00000070   28 28 28 28 28 28 28 28  ((((((((
00000078   28 28 28 28 28 28 28 28  ((((((((
00000080   28 28 28 28 28 28 28 28  ((((((((
00000088   28 28 28 28 28 28 28 28  ((((((((
00000090   28 28 28 28 28 28 28 28  ((((((((
00000098   28 28 28 28 28 28 FF C0  ((((((ÿÀ
000000A0   00 11 08 01 40 00 F0 03  ....@.ð.
000000A8   01 22 00 02 11 01 03 11  ."......
```

```
C:\Users\ASUS\STEGANOGRAPHY\output_with_exif.jpg

Offset(h)  00 01 02 03 04 05 06 07  Decoded text
00000000   FF D8 FF E0 00 10 4A 46  ÿØÿà..JF
00000008   49 46 00 01 01 01 00 60  IF.....`
00000010   00 60 00 00 35 39 35 39  .`..5959
00000018   31 30 33 32 48 6E 53 76  1032HnSv
00000020   33 6A 4A 69 4C 46 65 73  3jJiLFes
00000028   35 55 33 52 69 49 5A 50  5U3RiIZP
00000030   44 37 31 75 5A 4D 62 72  D71uZMbr
00000038   57 4A 31 48 78 4D 56 78  WJ1HxMVx
00000040   32 74 2B 79 69 42 73 3D  2t+yiBs=
00000048   FF DB 00 43 00 06 04 05  ÿÛ.C....
00000050   06 05 04 06 06 05 06 07  ........
00000058   07 06 08 0A 10 0A 0A 09  ........
00000060   09 0A 14 0E 0F 0C 10 17  ........
00000068   14 18 18 17 14 16 16 1A  ........
00000070   1D 25 1F 1A 1B 23 1C 16  .%...#..
00000078   16 20 2C 20 23 26 27 29  . , #&')
00000080   2A 29 19 1F 2D 30 2D 28  *)..-0-(
00000088   30 25 28 29 28 FF DB 00  0%()(ÿÛ.
00000090   43 01 07 07 07 0A 08 0A  C.......
00000098   13 0A 0A 13 28 1A 16 1A  ....(...
000000A0   28 28 28 28 28 28 28 28  ((((((((
000000A8   28 28 28 28 28 28 28 28  ((((((((
```

Figure 5. Input and output JPEG files and message insertion view

Base64 Decoding and AES-256 Decryption Process in ECB Mode (Extraction)

The extraction process is the reverse of the embedding process. The first step is to identify the "5959" marker at the end of the EXIF. This marker indicates that the metadata contains valid encrypted ciphertext. If the marker is detected in the image file used as input, the marker is separated from the actual data, namely the ciphertext in Base64 format. The ciphertext data is then decoded before the decryption process.

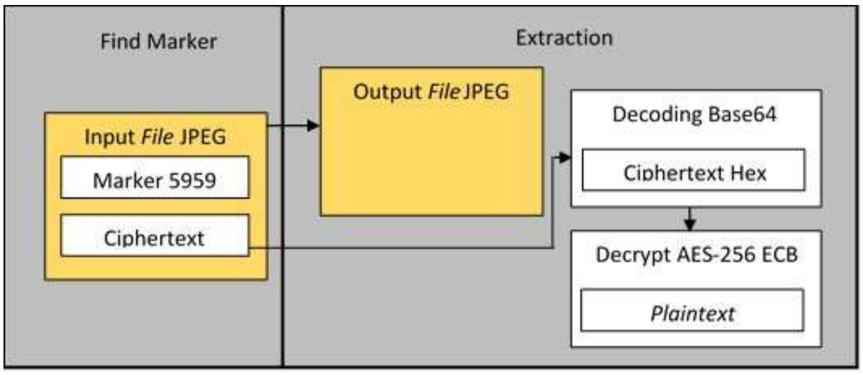


Figure 6 Message extraction process in the image

The following is the message extraction and decryption process from the image above:

1. Extract EXIF metadata from the JPEG file.
2. Check for the presence of the "5959" marker.
3. Separate the ciphertext from the marker and convert it to Base64 → Byte format.
4. Decrypt with AES-256 ECB using the correct key.
5. Convert the decrypted bytes into the original message string.
6. Decrypt using the AES-256 key.

The ciphertext, converted back to bytes, is processed using the AES-256 ECB algorithm using the same key used for encryption. Before decryption, the ciphertext length is checked to ensure it is a multiple of the AES block (16 bytes). After the decryption process is complete, the resulting byte data is then stripped of padding and converted back into a string to obtain the original message.

CONCLUSION

Based on the results and testing conducted, it can be concluded that the method of hiding secret messages using AES-256 encryption embedded in JPEG EXIF metadata provides an effective combination of security and simplicity of implementation. The message can be successfully hidden without changing the visual appearance of the image file, which confirms the main principle of steganography: imperceptibility. The use of ECB mode, although having weaknesses in the context of repetitive data, is quite adequate for short, unpatterned text, especially with the addition of cryptographically strong 256-bit encryption. The system's success in scenarios with intact or only lightly modified metadata proves this method is suitable for confidential communications that do not go through social media or platforms that automatically remove metadata. However, weaknesses arise in scenarios where EXIF metadata is completely removed or files are converted to other formats such as PNG, because the hidden message is lost along with the metadata. This indicates that the system's robustness to destructive environments is still limited and can be a point of interest for further research. Furthermore, encryption and decryption from a performance perspective using the Crypto.Cipher.AES library are fast and lightweight. Metadata insertion is also stable, making the system technically feasible for use in the context of lightweight desktop applications. Additionally, the presence of the special marker "5959" provides additional benefits in terms of detection efficiency and message extraction, although it should be noted that this method is not yet equipped with integrity validation to ensure that the data has not been modified.

REFERENCE

[1] F. Şahin, T. Çevik, and M. Takaoğlu, "Review of the Literature on the Steganography Concept," 2021.

[2] K. D. Michaylov and D. K. Sarmah, "Steganography and steganalysis for digital image enhanced Forensic analysis and recommendations," *Journal of Cyber Security Technology*, pp. 1–27, Jan. 2024, doi: 10.1080/23742917.2024.2304441.

[3] A. A. Pekerti, A. Sasongko, and A. Indrayanto, "Secure End-to-End Voice Communication: A Comprehensive Review of Steganography, Modem-Based Cryptography, and Chaotic Cryptography Techniques," *IEEE Access*, vol. 12, pp. 75146–75168, 2024, doi: 10.1109/ACCESS.2024.3405317.

[4] S. A. Sheik and A. P. Muniyandi, "Secure authentication schemes in cloud computing with glimpse of artificial neural networks: A review," Dec. 01, 2023, *KeAi Communications Co.* doi: 10.1016/j.csa.2022.100002.

[5] Rizki Maulana, "Capture The Flag: Cara Seru Belajar Cyber Security," Dicoding Indonesia. Accessed: May 02, 2025. [Online]. Available: https://www.dicoding.com/blog/capture-the-flag-cara-seru-belajar-cyber-security/

[6] J. M. Ghazali, S. M. N. Khan, and L. Q. Zakaria, "Image classification using EXIF metadata," *International Journal of Engineering Trends and Technology*, no. 1, pp. 69–73, Aug. 2020, doi: 10.14445/22315381/CATI3P211.

[7] P. Harvey, "JPEG Tags," ExifTool. Accessed: Mar. 02, 2025. [Online]. Available: https://exiftool.org/TagNames/JPEG.html

[8] Accusoft, "JPEG Metadata Structure," Accusoft Corporation. Accessed: Mar. 02, 2025. [Online]. Available: https://help.accusoft.com/ImageGear-Net/v25.0/Windows/HTML/JPEG_Non-Image_Data_Structure.html

[9] S. Sunardi, I. Riadi, and M. H. Akbar, "Steganalisis Bukti Digital pada Media Penyimpanan Menggunakan Metode Static Forensics," *Jurnal Nasional Teknologi dan Sistem Informasi*, vol. 6, no. 1, pp. 1–8, May 2020, doi: 10.25077/teknosi.v6i1.2020.1-8.

[10] A. B. L Nikhitha V S Arjun Naveen Chandra Gowda, "Survey of applications, advantages, and comparisons of AES encryption algorithm with other standards," *International Journal of Computational Learning & Intelligence*, vol. 2, no. 2, pp. 87–98, Mar. 2023, [Online]. Available: www.milestoneresearch.in

[11] P. S. Curlin, J. Heiges, C. Chan, and T. S. Lehman, "A Survey of Hardware-Based AES SBoxes: Area, Performance, and Security," *ACM Comput Surv*, vol. 57, no. 9, pp. 1–37, Sep. 2025, doi: 10.1145/3724114.

[12] F. L. De Mello and J. A. M. Xexéo, "Identifying Encryption Algorithms in ECB and CBC Modes Using Computational Intelligence," *Journal of Universal Computer Science*, vol. 24, no. 1, pp. 25–42, 2018.

[13] Z. Alimzhanova, M. Skublewska-Paszkowska, and D. Nazarbayev, "Periodicity Detection of the Substitution Box in the CBC Mode of Operation: Experiment and

Study," *IEEE Access*, vol. 11, pp. 75686–75695, 2023, doi: 10.1109/ACCESS.2023.3295909.

[14] N. M. Ansari *et al.*, "A Review: Importance, Implementation, and Enhancement of AES," *Journal of Xi'an Shiyou University, Natural Science Edition*, vol. 18, no. 8, pp. 8–15, Aug. 2022, [Online]. Available: http://xisdxjxsu.asia

[15] S. Ahmed *et al.*, "Lightweight AES Design for IoT Applications: Optimizations in FPGA and ASIC with DFA Countermeasure Strategies," *IEEE Access*, 2025, doi: 10.1109/ACCESS.2025.3533611.

[16] Wikipedia, "Golomb coding." Accessed: Mar. 02, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Golomb_coding

[17] E. P. Stoilov, "Discovery And Analysis of Exif Data in Images," in *Proceedings Of University Of Ruse - 2022*, Ruse, Bulgaria: Angel Kanchev" University of Ruse, 2022, pp. 52–59. Accessed: May 02, 2025. [Online]. Available: https://conf.uni-ruse.bg/bg/docs/cp22/3.2/3.2-31.pdf

[18] H. Wijayanto, I. Riadi, and Y. Prayudi, "Encryption EXIF Metadata for Protection Photographic Image of Copyright Piracy," *Hendro Wijayanto et al, IJRCCT*, vol. 5, no. 5, 2016, [Online]. Available: www.ijrcct.org

[19] F. Fatima and E. Azeem, "Data Carving - The Art of Retrieving Deleted Data as Evidence," *International Journal for Electronic Crime Investigation*, vol. 6, no. 2, pp. 23–32, 2022.

[20] R. Indrayani, P. Ferdiansyah, and M. Koprawi, "Analisis Penggunaan Kriptografi Metode AES 256 Bit pada Pengamanan File dengan Berbagai Format," *Digital Transformation Technology*, vol. 4, no. 2, pp. 1245–1251, Feb. 2025, doi: 10.47709/digitech.v4i2.5457.

[21] R. S. Bayu Adi, "Comparative Analysis of AES-Turbo Code Combination Encryption Method on Three Variations AES Key," *International journal of science, engineering, and information technology(IJSEIT)*, vol. 2, no. 1, pp. 53–55, Jul. 2017, [Online]. Available: https://journal.trunojoyo.ac.id/ijseit

[22] T. Wu, X. Hu, and C. Liu, "Security-oriented steganographic payload allocation for multi-remote sensing images," *Sci Rep*, vol. 14, no. 1, Dec. 2024, doi: 10.1038/s41598-024-55474-y.

[23] W. Simoes and M. de Sá, "PSNR and SSIM: Evaluation of the Imperceptibility Quality of Images Transmitted over Wireless Networks," in *Procedia Computer Science*, Elsevier B.V., 2024, pp. 463–470. doi: 10.1016/j.procs.2024.11.134.

[24] H. Karajeh and M. Maqableh, "An imperceptible, robust, and high payload capacity audio watermarking scheme based on the DCT transformation and Schur decomposition," *Analog Integr Circuits Signal Process*, vol. 99, no. 3, pp. 571–583, Jun. 2019, doi: 10.1007/s10470-018-1332-0.

[25] T. Beyne, Y. L. Chen, and M. Verbauwhede, "A Robust Variant of ChaCha20-Poly1305," *Cryptology ePrint Archive*, 2025, Accessed: May 01, 2025. [Online]. Available: https://eprint.iacr.org/2025/222

[26] P. Dharshini, J. Arokia Renjith, and P. Mohan Kumar, "Screening the covert key using

honey encryption to rule out the brute force attack of AES—a survey," Dec. 01, 2016, *John Wiley and Sons Inc.* doi: 10.1002/sec.1753.

[27] P. Manikandaprabhu and M. Samreetha, "A Review of Encryption and Decryption of Text Using the AES Algorithm."

[28] W. Muła and D. Lemire, "Faster Base64 Encoding and Decoding Using AVX2 Instructions," Jun. 2018, doi: 10.1145/3132709.