

Marker-based crypto-steganographic system for secure data transmission over open channels

Denys Odnoochko^{1,†}, Anna Ilyenko^{1,*,†}, Sergii Ilyenko^{1,†}, Oleksii Smirnov^{2,†} and Hennadii Hulak^{3,†}

¹Scientific Cyber Security Association of Ukraine, Mykhaila Dontsia Str., 2A, Kyiv, 03161, Ukraine

²Central Ukrainian National Technical University, University Ave., 8, Kropivnitskiy, 25006, Ukraine

³Borys Grinchenko Kyiv Metropolitan University, Bulvarno-Kudriavska Str., 18/2, Kyiv, 04053, Ukraine

Abstract

This study analyses contemporary cryptographic and steganographic approaches to information security. During the analysis of symmetric and asymmetric cryptographic algorithms in combination with spatial and frequency-domain steganographic methods, it was observed that the use of a hybrid crypto-steganographic approach produces a synergistic effect, resulting in improved overall information protection compared to the use of individual mechanisms. Based on this analysis, a novel solution for implementing multi-level protection of confidential information transmitted over open communication channels is proposed. The proposed approach is based on the AES block cipher with optional RSA and ECC algorithms for symmetric key encapsulation, combined with a steganographic mechanism that utilises a marker embedded after the main content of the container. As part of the practical contribution, the proposed algorithm was implemented as a software module in the Python programming language. The developed crypto-steganographic system is versatile, as it enables secure encryption and concealment of data across a wide range of file types and formats. The obtained results may be applied in both scientific research and real-world information and communication systems.

Keywords

information protection, cryptography, steganography, encryption, crypto-steganographic system, cryptographic analysis, steganographic analysis

1. Introduction

In the context of rapid technological progress, the exchange of information has become somewhat easier. For the vast majority of people, sharing confidential data in the digital realm is an integral part of everyday life. At the same time, the development of information technology is accompanied by a rise in the number of potential threats, creating the conditions for data leaks and unauthorised access to information.

Today, the problem of data leaks when using open communication channels is becoming particularly acute due to the human factor, as many people, due to a lack of awareness or carelessness, fail to use additional security measures when transmitting sensitive data, which constitutes a direct breach of the confidentiality, integrity and availability of information, as it becomes vulnerable to interception and modification. Figure 1 shows the results of a study [1] conducted by the German online platform Statista, which specialises in the collection and visualisation of statistical data. As of February 2024, 46% of companies worldwide had experienced a leak of confidential information.

Leaks of confidential information can lead to fraud, identity theft and the manipulation of public opinion, as well as having psychological and social consequences for those affected. With this in mind, confidential information must be treated as a valuable asset that requires proper protection.

WCH'26: Fifth International Workshop on Cyber Hygiene, June 25–26, 2026, Krakow, Poland

*Corresponding author.

†These authors contributed equally.

✉ remixr@ukr.net (D. Odnoochko); anna.ilyenko@npp.kai.edu.ua (A. Ilyenko); serhii.ilyenko@npp.kai.edu.ua (S. Ilyenko); dr.smirnova@gmail.com (O. Smirnov); h.hulak@kubg.edu.ua (H. Hulak)

ORCID 0009-0004-1013-5766 (D. Odnoochko); 0000-0001-8565-1117 (A. Ilyenko); 0000-0002-0437-0995 (S. Ilyenko); 0000-0001-9543-874X (O. Smirnov); 0000-0001-9131-9233 (H. Hulak)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Share Of Organizations Worldwide That Have Experienced A Loss Of Sensitive Information As Of February 2024, By Country

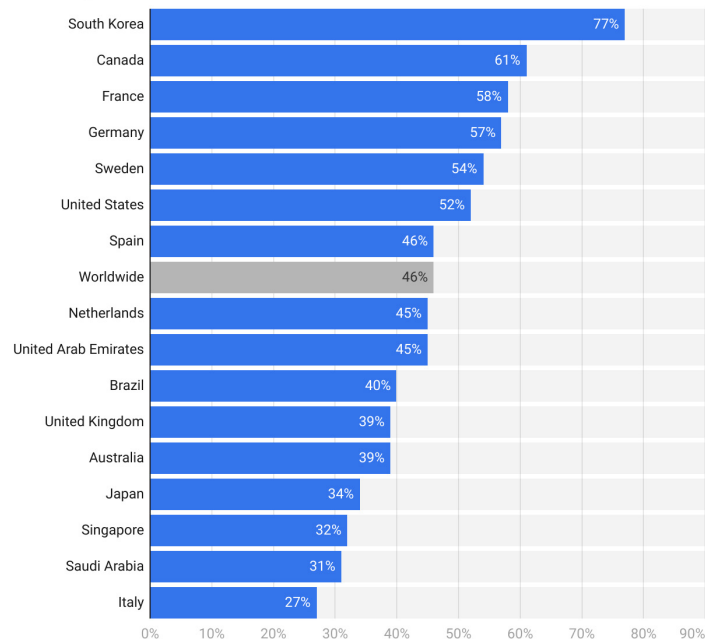


Figure 1: Statistics on data breaches.

To preserve the confidentiality of information when transmitting it over open communication channels, two methods of software protection are traditionally used: cryptographic and steganographic methods. However, it is worth noting that cryptography and steganography are usually applied separately, as they employ different approaches to ensuring information security. However, the separate use of either method has certain limitations. The use of cryptography alone can ensure an adequate level of protection for the message content through encryption, and also enables the implementation of authentication and data integrity verification mechanisms, in particular through the creation of an electronic digital signature. However, cryptographic protection does not conceal the very fact that encrypted information is being transmitted, which may attract the attention of a potential attacker and provide grounds for cryptanalysis. In turn, the use of steganography alone, whilst allowing the fact of data transmission to be concealed by embedding it covertly into media files, means that if a hidden message is detected during steganographic analysis, it will be revealed in the absence of additional encryption. Consequently, neither approach on its own provides a comprehensive level of information security, which justifies combining them.

In accordance with the defined objectives for information protection, the combination of cryptographic algorithms with steganographic methods is applied in several conceptually different approaches: the protection of intellectual property rights or the covert exchange of information.

The first approach is based on the use of a specialised crypto-steganographic solution to generate a digital watermark for the protection of the user's property and non-property rights prior to the distribution of media content. In this case, the user employs a secret identifier, on the basis of which a fixed-length bit sequence (hash) is generated using cryptographic hash functions, which is then embedded into the media file via steganography. As a result, the watermark applied to the media file is invisible to the human eye, but can be easily verified using the same crypto-steganographic solution, whereby the generated hash of the secret identifier is compared with the embedded hash in the media file.

The second approach focuses on ensuring the confidentiality of sensitive data when transmitted over unsecured communication channels. Under this approach, the first stage involves the use of

a specialised crypto-steganographic solution to encrypt sensitive information using cryptographic algorithms, thereby ensuring that it cannot be read without knowledge of the key. The ciphertext is then embedded into a container using a steganographic method. In this way, steganography conceals the very fact that sensitive information is being transmitted, whilst cryptography protects its content.

Although combining cryptography with classical steganography methods provides comprehensive information protection, it has certain practical limitations, as the quality of the transmitted steganographic container depends directly on the size of the hidden message. Consequently, there is a problem regarding the optimal distribution of hidden data within the container. In this regard, when developing a crypto-steganographic solution, the priority is to find adaptive embedding methods, the use of which will ensure the transmission of a significant volume of hidden information without significant distortion of the "cover".

The purpose of this article is to investigate cryptographic algorithms and steganographic approaches for combined information protection, to analyse the principles underlying the design of crypto-steganographic systems, and to develop a novel approach for ensuring multi-level protection of information resources. The proposed approach is based on the use of the AES block cipher operating in GCM mode in combination with the asymmetric RSA and ECC algorithms for symmetric key encapsulation, as well as a marker-based steganographic mechanism for concealing encrypted data within multimedia containers. The developed approach aims to enhance the confidentiality, integrity, and concealment of sensitive information transmitted over open communication channels while maintaining compatibility with various media file formats.

Based on the purpose, the task of this scientific article:

- to investigate modern cryptographic and steganographic approaches.
- to develop an algorithm for combined information protection using cryptographic algorithms in conjunction with a marker-based steganographic method.
- to implement a software module for combined protection using the Python programming language.
- to test and evaluate the effectiveness of the developed solution, and compare it with existing similar systems.

In this research paper, cryptographic protection is based on the use of a symmetric encryption algorithm for data confidentiality and an asymmetric cryptographic scheme for encapsulating the symmetric key. Steganographic protection is based on the use of a dedicated marker appended after the main content of media files, which enables the reliable concealment of encrypted data and supports efficient extraction and decryption of the ciphertext during the unpacking of the steganographic container.

This paper proposes an integrated approach combining cryptographic and steganographic mechanisms, the structural implementation of which enables the covert transmission of confidential information without modifying the primary content of media files. The proposed approach addresses the problem of optimal placement of hidden data within the container, as its embedding does not introduce distortions into the media content that could be detected by conventional steganalysis techniques.

The practical significance of the study lies in the design and implementation of a crypto-steganographic software module in Python, utilising modern cryptographic libraries and multimedia processing tools for secure data encryption and handling. The developed algorithm and software module can be applied in information security systems and in scientific research in the fields of cryptography, steganography, and cybersecurity. The proposed approach enhances the level of information security during data transmission over unsecured communication channels.

2. References survey and problem statement

In modern information and communication systems, cryptographic methods play a dominant role in ensuring data security; these can be divided into two main types, namely symmetric and asymmetric

algorithms. However, when it comes to ensuring the confidentiality of information by converting it into an unreadable form, the use of only one of these types does not always allow for an optimal balance between the level of security and system performance. This is precisely why combined approaches, which offer the advantages of both symmetric and asymmetric cryptographic algorithms, are becoming increasingly widespread.

An analysis of current scientific research suggests that time-tested and standardised algorithms are most commonly used in the design of hybrid cryptosystems. In particular, based on the conclusions in the articles [2, 3, 4], which were devoted to the comparison of symmetric algorithms, AES can be characterised as the optimal option in terms of the balance between cryptographic strength, speed and efficiency of computational resource utilisation. Among public-key algorithms, RSA and ECC occupy leading positions, as noted in the articles [5, 6, 7]. The results of the comparative studies described in these articles demonstrate that ECC has significant advantages over RSA in terms of energy efficiency, memory consumption and overall performance, particularly under resource-constrained conditions.

Therefore, given current trends in the development of cryptosystems, and with a view to ensuring the rapid encryption of large volumes of data and the secure distribution of keys, it is advisable to conduct further research into hybrid approaches based on combining AES with RSA or AES with ECC.

In article [8], the authors described the implementation of a combined cryptographic approach based on the "envelope" principle, whereby the plaintext is first encrypted using the AES algorithm, after which the secret AES key is further encrypted using RSA before the information is transmitted. The proposed algorithm was simulated on Xilinx ISE 14.7 using VHDL.

In article [9], the authors described a software implementation of a hybrid cryptosystem in the Python programming language, which combines AES-GCM with RSA-OAEP. Furthermore, the paper presents benchmarking results for the cryptosystem across various file sizes and RSA key lengths, confirming the validity of the proposed approach.

In article [10], the author conducted a comparative analysis of two hybrid cryptographic approaches, namely AES with ECC and DES with ECC, with the aim of assessing their performance and practical suitability for processing different types of data. The results of the study show that the combination of ECC with AES provides higher security and faster encryption and decryption speeds compared to ECC+DES, particularly when working with large volumes of data, such as videos, documents and other files.

In article [11], the authors propose a combined approach based on the implementation of the ECDH key exchange mechanism, where security is ensured by elliptic curve point multiplication operations, whilst AES is used for the direct encryption of information. The proposed methodology is designed for use in secure cloud-based healthcare systems, where both performance and a high level of protection for confidential data are critical.

In article [12], the authors propose a new concept for a hybrid cryptographic algorithm that combines AES, RSA and the Elliptic Curve Integrated Scheme (ECIES) for the secure transmission of information from one device to another.

Although the articles cited discuss effective combined cryptographic approaches, they focus primarily on protecting the content of the information, whilst the aspect of concealing the very fact of its transmission remains no less important. Although steganographic methods are used much less frequently than classical cryptographic solutions, they remain relevant in cases where it is necessary not only to ensure data confidentiality but also to conceal the very fact of its transmission. This is precisely why researchers are developing hybrid variants of information security based on the integration of cryptographic methods with steganographic approaches.

When implementing hybrid crypto-steganographic solutions, the LSB (Least Significant Bit) method is most commonly used among the various steganographic approaches for working with the vast array of media files. One of the well-known approaches, discussed in article [13], involves integrating AES-128 and SHA-256 using the LSB method into an HEVC (High Efficiency Video Coding) video stream. The developers propose using a single key that is subjected to a hashing process using SHA-256. The resulting hash values are used as a key for encryption and to determine the positions for embedding the data. A similar method is described in the scientific article [14], although it uses DES with a fixed

key, and audio files serve as the container. A common drawback of these solutions is that ensuring the accuracy of transmitted data is not the primary objective, as maintaining a constant bitrate during the transmission of audio or video streams remains the priority. Consequently, errors in frames are not corrected by retransmission, which creates a risk of damage to the embedded information. Encryption in such cases makes it impossible to recover lost data. Furthermore, applying the LSB method to HEVC formats and audio files can cause significant distortion of the embedded data, which potentially reduces the effectiveness of this approach.

In article [15], the authors propose a method in which confidential data is encrypted using the AES algorithm in OFB mode, after which the encryption key is appended to the encrypted message, and the entire packet is concealed within an image file by clustering the nearest centroid. However, storing the key together with the encrypted data and subsequently transmitting them in a single container constitutes a critical vulnerability. In the event of a successful steganographic analysis, an attacker could easily obtain both the encrypted data and the key to it, thereby negating the benefits of the cryptography used.

The DCT (Discrete Cosine Transform) and DWT (Discrete Wavelet Transform) are used to work with the frequency domain of media files. In article [16], the authors propose a crypto-steganographic framework for the secure transmission of medical data, which combines the ChaCha20 stream cipher with ECC for key security, whilst information hiding is implemented using 2D discrete wavelet transform (DWT). And in article [17], another interesting crypto-steganographic approach for image hiding is proposed, in which the secret image is embedded into a container image using a combination of several algorithms. In particular, DWT is used to process the container, and DCT is used to compress the hidden image in order to reduce its size and improve embedding efficiency. Additionally, to enhance security, the RSA algorithm is used to encrypt the secret image before it is hidden, thereby increasing the system's resistance to unauthorised access and attacks.

Although the use of classical spatial domain methods allows for the embedding of a significant amount of hidden data (up to 40%), this can lead to noticeable visual distortion. In contrast, approaches based on frequency transformations, whilst resistant to media file compression, typically have limited capacity (approximately 15–20%) and are used primarily for embedding digital watermarks rather than for concealing significant amounts of data. Furthermore, neither spatial nor frequency-based methods are sufficiently resistant to modern methods of steganographic analysis.

3. Architecture of the proposed crypto-steganographic module

The aim of the study is to develop a module for robust cryptographic and steganographic information protection, the operating principle of which is based on the use of the AES symmetric algorithm for rapid data encryption prior to concealing it within media files using a steganographic mechanism that operates by embedding a marker for retrieval. Since the structural implementation involves embedding the symmetric algorithm key together with the hidden message in the container, asymmetric RSA and ECC algorithms (as appropriate) are additionally used to protect it. This comprehensive approach not only solves the problem of transmitting information over open communication channels but also ensures secure key exchange, as it all comes down to the use of a key pair during the embedding or extraction of data.

Functional requirements:

- When using the window add-on, users can select files in various formats for further processing and hiding.
- To encrypt the required file, the user can select the key size for the symmetric algorithm, which will be automatically generated during the operation of the crypto-steganographic module.
- The user can create a key pair in accordance with the type of asymmetric algorithm and the key sizes selected from those available.
- To encapsulate the symmetric algorithm key, the user can specify the path to the recipient's public key.

System requirements: the software module is being developed as a Windows application in EXE format. The following software development tools were selected for the implementation of the crypto-steganographic protection module: 1) development environment – Visual Studio Code; 2) programming language: Python; 3) graphical user interface development environment: Qt Designer (PyQt5).

The crypto-steganographic module operates in accordance with the block diagram (Fig. 2), in which information is processed within the blocks shown.

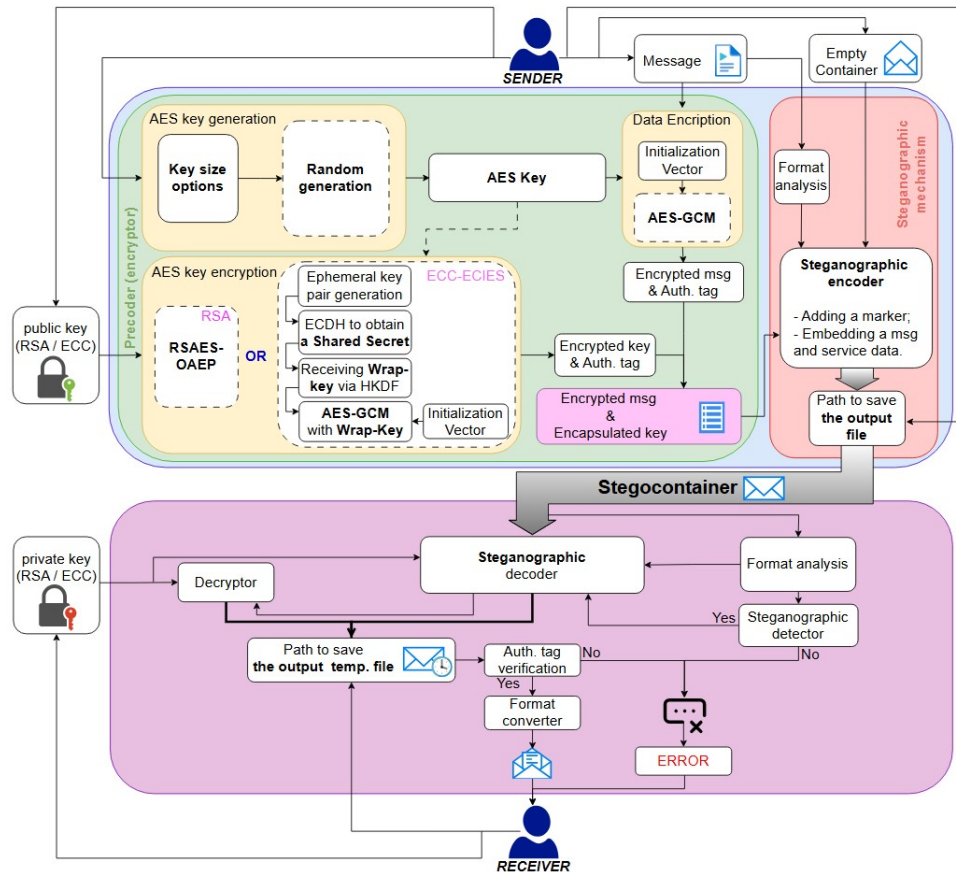


Figure 2: Block diagram of the crypto-steganographic module.

Before using the crypto-steganographic module for covert communication between two users, the recipient must generate a pair of PEM-format keys for the selected asymmetric algorithm and transmit the public key to the sender via open communication channels. On the sender's side, the process of creating a steganographic container begins by copying the selected media file, as the container, to the specified save path for the source file. Before using this container, the information to be concealed is prepared. Such information can be placed in any file, so the message format is read.

Cryptographic data processing takes place in the cipher block – the precoder. The AES symmetric block cipher algorithm was selected for message encryption, specifically in Galois/Counter Mode (GCM). Its distinctive feature is the combination of stream cipher encryption in counter mode (CTR) with an authentication mechanism. By using GCM, which has been standardised by NIST, it is possible not only to ensure the confidentiality of information during transmission, but also to verify its integrity.

The authentication mechanism in GCM mode is based on a hash function called GHASH, which utilises multiplication by a hash key in the finite Galois field. The hash key, denoted as H , is generated by applying a block cipher to a zero block. The GHASH function is a key hash function, but is not in itself a cryptographic hash function. It is based on multiplication in the field $GF(2^{128})$ with the irreducible polynomial $F(x) = x^{128} + x^7 + x^2 + x + 1$ [18].

The encryption of information using the AES cryptographic algorithm in GCM mode can be formulated mathematically. Let n and u be a unique pair of positive integers such that the total number

of bits in the plaintext is equal to $(n - 1)128 + u$, where $1 \leq u \leq 128$. The plaintext consists of a sequence of n -bit strings, in which the length of the last string is u , and the length of all other strings is 128 bits. This sequence is denoted as $P_1, P_2, \dots, P_{n-1}, P_n^*$, and these bit strings are called data blocks, although the last string P_n^* may be an incomplete block. Similarly, the ciphertext is denoted as $C_1, C_2, \dots, C_{n-1}, C_n^*$, where the number of bits in the last block C_n^* is equal to u . Additional authenticated data A is denoted as $A_1, A_2, \dots, A_{m-1}, A_m^*$, where the last bit string A_m^* may be a partial block of length v , and m and v are a unique pair of positive integers such that the total number of bits in A is $(m - 1)128 + v$, where $1 \leq v \leq 128$ [19].

The first stage of the authenticated encryption operation is the generation of the hash key H , which is determined by formula (1), where E_K is the AES block cipher with key K :

$$H = E_K(0^{128}) \quad (1)$$

The initial value of the counter Y_0 is then calculated using equation (2), which depends on the length of the initialisation vector (IV):

$$Y_0 = \begin{cases} IV || 0^{31} || 1 & \text{if } \text{len}(IV) = 96 \\ GHASH(IV || 0^s || 0^{64} || \text{len}_{64}(IV)) \text{ with } s = 128 - \text{len}(IV) \bmod 128 & \text{otherwise} \end{cases} \quad (2)$$

The initialisation vector is a nonce used in authenticated plaintext encryption with additional authentication data (AAD) that is not encrypted.

Once the initial value Y_0 has been determined, the process moves on to generating the sequence of counters that form the basis of the CTR mode. To do this, the recursive relation (3) is used, where the increment function interprets the rightmost 32 bits as an integer and increments it by one modulo 2^{32} :

$$Y_i = \text{incr}(Y_{i-1}) \quad \text{for } i = 1, \dots, n \quad (3)$$

In this way, a sequence of pseudo-random input values is generated for the block cipher. The next step involves the actual encryption of the plaintext. Using formula (4), the operation is performed for each block P_i , where $i = 1, \dots, n - 1$:

$$C_i = P_i \oplus E_K(Y_i) \quad (4)$$

This operation follows the classic stream cipher scheme, in which the result of applying AES to the counter value acts as the ciphertext stream. For the latest, possibly incomplete block P_n^* , only the required number of most significant bits is used:

$$C_n^* = P_n^* \oplus \text{MSB}_u(E_K(Y_n)) \quad (5)$$

Thus, the complete ciphertext C is formed in this way; however, to verify data integrity, it is necessary to generate a specific checksum for authenticity verification. For this purpose, a special function GHASH is used, defined by formula (6), where A is the AAD:

$$\text{GHASH}(H, A, C) = X_{m+n+1} \quad (6)$$

The variables X_i for $i = 0, \dots, m + n + 1$ are defined as follows [19]:

$$X_i = \begin{cases} 0 & \text{for } i = 0 \\ (X_{i-1} \oplus A_i) \cdot H & \text{for } i = 1, \dots, m - 1 \\ (X_{m-1} \oplus (A_m^* || 0^{128-v})) \cdot H & \text{for } i = m \\ (X_{i-1} \oplus C_i) \cdot H & \text{for } i = m + 1, \dots, m + n - 1 \\ (X_{m+n-1} \oplus (C_n^* || 0^{128-u})) \cdot H & \text{for } i = m + n \\ (X_{m+n} \oplus (\text{len}(A) || \text{len}(C))) \cdot H & \text{for } i = m + n + 1 \end{cases} \quad (7)$$

Once the GHASH value has been obtained, the algorithm proceeds to the final step – generating the authentication tag. To do this, the GHASH result is combined with the encrypted initial counter value using formula (8), where T is a bit string of length t , formed by selecting the t most significant (leftmost) bits:

$$T = \text{MSB}_t(\text{GHASH}(H, A, C) \oplus E(K, Y_0)) \quad (8)$$

The AES key is encapsulated using the recipient's public key, which the sender uses alongside other input data to create a steganographic container. The public key is part of a cryptographic key pair; for its generation, the sender can choose between a 2048-bit RSA key or an elliptic curve-based algorithm (secp256r1, secp384r1).

When selecting the RSA algorithm, key generation will be carried out in the following stages [20]:

- Two large prime numbers p and q are chosen.
- The modulus is calculated using the following formula (9):

$$n = p \cdot q \quad (9)$$

- The Euler function is calculated using formula (10):

$$\varphi(n) = (p - 1)(q - 1) \quad (10)$$

- The public exponent is chosen using formula (11):

$$e : \text{gcd}(e, \varphi(n)) = 1 \quad (11)$$

- The private exponent is calculated using formula (12):

$$d \equiv e^{-1} \pmod{\varphi(n)} \quad (12)$$

Consequently, the public key is $K_{pub} = (e, n)$, and the private key is $K_{priv} = (d, n)$.

Direct encryption of the AES key on the sender's side is performed using the Optimal Asymmetric Encryption Padding (OAEP) scheme, which ensures resistance to adaptive attacks. Let the message M (in this case, the AES key) have a length less than $k - 2hLen - 2$, where k is the module length (256 bytes) and $hLen$ is the hash length, which is 32 bytes since the SHA-256 hash function is used. In that case, the RSAES-OAEP algorithm works as follows [21]:

- A hash for the symmetric key is generated using formula (13), where L is an auxiliary parameter (label), usually an empty string:

$$lHash = H(L) \quad (13)$$

- To ensure the specified size, a sequence of zero PS bytes is generated using formula (14), where $mLen$ is the length of the plaintext (32 bytes):

$$PS = k - mLen - 2hLen - 2 \quad (14)$$

- Next, according to formula (15), a DB block is created, which is intermediate and contains the message M together with service data:

$$DB = lHash \parallel PS \parallel 0x01 \parallel M \quad (15)$$

- A random value is then generated:

$$seed \in \{0, 1\}^{hLen} \quad (16)$$

- Next, the mask generation function MGF is used, which is based on a hash function and is calculated as follows:

$$\begin{aligned} dbMask &= MGF(seed, k - hLen - 1) \\ maskedDB &= DB \oplus dbMask \\ seedMask &= MGF(maskedDB, hLen) \\ maskedSeed &= seed \oplus seedMask \end{aligned} \quad (17)$$

- This results in an encoded message, which takes the following form:

$$EM = 0x00 || maskedSeed || maskedDB \quad (18)$$

- In the final stage, standard RSA encryption is performed using formula (19):

$$C = EM^e \bmod n \quad (19)$$

If, however, an algorithm based on elliptic curves is chosen, the procedure for generating a key pair and encapsulating the AES key will differ slightly. Such an algorithm is implemented using the ECIES and works as follows:

- One of the two possible elliptic curve variants (secp256r1, secp384r1) is selected in accordance with formula (20), where a, b are the parameters of the curve:

$$E : y^2 = x^3 + ax + b \pmod{p} \quad (20)$$

- Based on the selected elliptic curve, a permanent key pair is generated for the recipient using formula (21), where d_R is the recipient's private key, Q_R is the recipient's public key, G is the generator point of the curve, and n is the order of point G :

$$d_R \in [1, n - 1], \quad Q_R = d_R G \quad (21)$$

- A one-time key pair is automatically generated on the sender's side using formula (22):

$$d_E \in [1, n - 1], \quad Q_E = d_E G \quad (22)$$

- Next, a shared secret S is derived via ECDH using the recipient's public key, which is calculated using equation (23):

$$S = d_E Q_R \quad (23)$$

- The shared secret S is used to derive the symmetric key K using formula (24), where S is the shared secret and N is the length of the AES key:

$$K = HKDF(S, \text{info "ecies-wrap"}, N) \quad (24)$$

Using the symmetric key K , the sender encrypts the AES key by repeatedly applying the AES algorithm in GCM mode.

Next, the ciphertext, which consists of the encrypted message and the embedded key, enters the next block – the steganographic mechanism, where, using a steganographic encoder, the ciphertext is embedded into the container after a corresponding marker, which consists of two parts – “STEGO-” and the “message format”, which was previously defined before encryption. In addition, identifiers such as the key size, encapsulation algorithm, curve tag and other service data are added to the structure of the steganographic container. As a result, the resulting steganographic container can be transmitted over open communication channels. Upon receiving the steganographic container, the recipient must select the previously generated private key to extract the hidden information.

From the recipient's perspective, the system begins by analysing the received file. During the analysis, the structure of the steganographic container is checked and a search is carried out for a special marker indicating the presence of hidden data. If no marker is found, the file is likely to be corrupted or does not contain any hidden information. In this case, the steganography detector will report an error. In turn, the steganography decoder is activated only upon confirmation of the presence of hidden information in the media file. The steganography decoder performs the process of extracting the hidden data. First, the service parameters are read from the container. After that, the encapsulated AES key is decrypted using the private key. If the private key belongs to a key pair of the RSA asymmetric algorithm, the decryption process is reversed [21]:

- First, the encoded message EM is calculated using formula (25):

$$EM = C^d \bmod n \quad (25)$$

- The encoded EM message is then broken down into its components using equation (26):

$$EM = 0x00 || maskedSeed || maskedDB \quad (26)$$

- Using the same MGF function, the following calculations are performed:

$$\begin{aligned} seedMask &= MGF(maskedDB, hLen) \\ seed &= maskedSeed \oplus seedMask \\ dbMask &= MGF(seed, k - hLen - 1) \\ DB &= maskedDB \oplus dbMask \end{aligned} \quad (27)$$

- The following components are extracted from the DB :

$$DB = lHash || PS || 0x01 || M \quad (28)$$

- The $lHash$ is then checked for validity, after which the zero bytes in the PS are removed and the M message (AES key) is restored.

If the private key belongs to a key pair of an asymmetric algorithm based on elliptic curves, decryption will be performed by computing the shared secret. Due to the properties of the elliptic curve algorithm, according to which $S = S' = d_E d_R G$, the decryption process will proceed as follows:

- On the recipient's side, following automatic extraction from the Q_E steganographic container, the secret key is calculated using formula (29):

$$S' = d_R Q_E \quad (29)$$

- Next, the HKDF is applied using the formula in (30) to generate a symmetric key, which will be used to decrypt the encapsulated key:

$$K = HKDF(S') \quad (30)$$

This results in the decryption of the AES key, which is then used to convert the ciphertext into plaintext. During this process, the integrity check is also verified to ensure that the steganographic container has not been altered prior to extraction.

The final stage involves the format converter. At this stage, the original extension of the hidden file is restored and the file is written to the file system at the specified path. As a result, the user receives the original file in its original form.

3.1. Software implementation of the crypto-steganographic module

When writing the code to create an application for crypto-steganographic information protection, modules from the standard Python library were used, such as os, shutil, time, threading, and traceback, which handle file system operations, organize the streaming processing of large data volumes, measure operation execution times, support multithreading, and handle errors. To implement modern and cryptographically secure encryption algorithms, the specialized cryptography library was used, which provides high-level and low-level primitives.

The algorithmic sequence of the software module's operation when embedding information into a container can be described as a sequence of interrelated stages combining cryptographic data processing and their subsequent steganographic embedding. The process of creating a steganographic container consists of 9 stages (Fig. 3): 1) Verification of configured parameters. 2) Initialization of parameters for the file embedding operation. 3) Generation of a random AES key. 4) Encapsulation of the AES key according to the selected asymmetric encryption algorithm. 5) Copying the container. 6) Adding the "STEGO-" marker and other service data. 7) Generating a nonce and initializing AES-GCM. 8) Encrypting the data and embedding it into the container. 9) Generating an authentication tag.

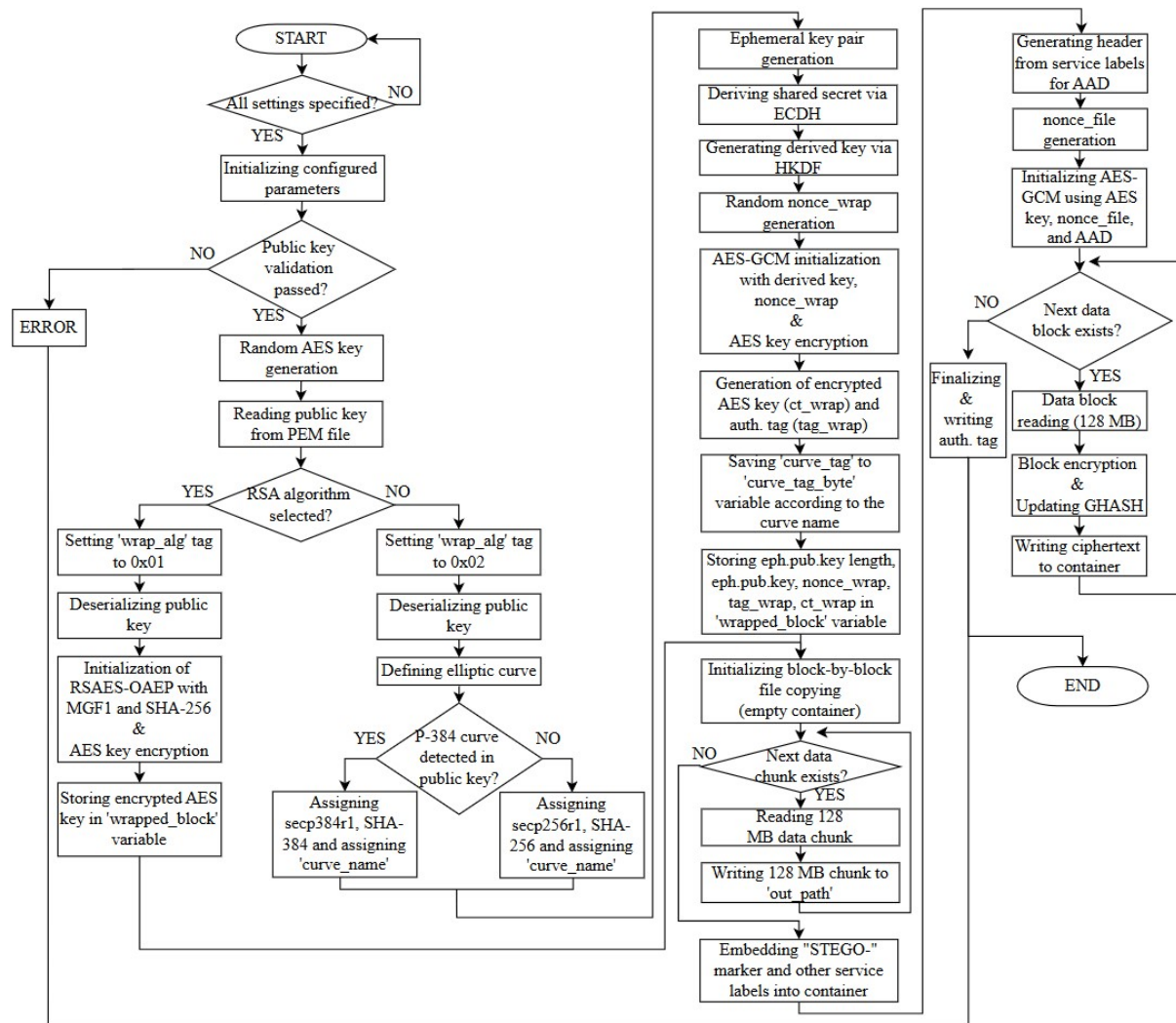


Figure 3: Algorithmic flowchart of the software module's operation during the creation of a steganographic container.

The algorithmic sequence of operations performed by the software module when extracting hidden information from a steganographic container is the reverse of the embedding process and combines the

stages of analysing the media file for the presence of embedded tags, cryptographic recovery of the encapsulated key, and subsequent decryption of the data with gradual extraction.

The process of extracting a message from a steganographic container consists of eleven stages:

- Checking the configured parameters.
- Initialising parameters for the extraction operation.
- Search for the “STEGO-” marker.
- Reading the extension of the hidden file.
- Reading the AES key size identifier.
- Reading the tag for the asymmetric algorithm and decapsulating the AES key.
- Reading the nonce.
- Locating the authentication tag at the end of the file.
- Initialising the AES-GCM algorithm.
- Step-by-step decryption and unpacking of the embedded message into a temporary file.
- Verifying the authentication tag.

At each stage, the service tags are checked for consistency, which allows us to monitor the integrity and accuracy of the process of extracting hidden information from a steganographic container. Only after the final verification of the authentication tag will the extension of the output file be changed from ".tmp" to the original one. If the authentication tag does not match, the temporary file will be deleted immediately and the extraction process will be terminated with an error.

3.2. Stego-container structure

The resulting steganographic container is an information object whose structure has been modified in such a way that it contains not only the source data (e.g. an image, audio or video file), but also an embedded hidden message along with the accompanying service elements.

The main requirement when creating a steganographic container is to preserve the natural characteristics of the media file used as the container. This means that the changes made must not result in noticeable visual, acoustic or statistical distortions that could indicate the presence of hidden information. In addition to invisibility, an important feature is the container's capacity, as the volume of embedded information affects the quality of the steganographic container. This is why examining the structure of the steganographic container is a necessary prerequisite for verifying that the crypto-steganographic module operates in accordance with the specified requirements.

Figure 4 illustrates the structure of the resulting steganographic container following the use of the developed crypto-steganographic module. To analyse the structure and identify all service elements, an .mp3 audio recording was used as the steganographic container, into which a hidden message in the form of a TXT-file was embedded.

Using a hex editor, we can see that the hidden section begins with a specially placed “STEGO-” marker at the end of the main content of the audio recording used. Accordingly, the hexadecimal characters for this marker are: 53 54 45 47 4F 2D. This marker plays a key role, as during extraction the algorithm is designed to locate the starting point of the embedded data by scanning the steganographic container from the very end.

The marker is followed by the service information “extension”, which indicates the format of the embedded message. Accordingly, the sequence of hexadecimal characters for the “txt” string is: 74 75 74, ending with the separator “|” (7C), followed by other service tags generated as a result of encryption and key encapsulation.

The first service tag is the identifier for the size of the symmetric AES key. It occupies one byte with the key length-selected when the steganographic container was created-encoded in binary form. In this case, the value 0x01 was added to the header, indicating an AES key size of 128 bits.

The second service tag is the next byte, which determines the value of "wrap alg", i.e. the type of asymmetric algorithm used to encapsulate the AES key. One of two options is provided for – RSA

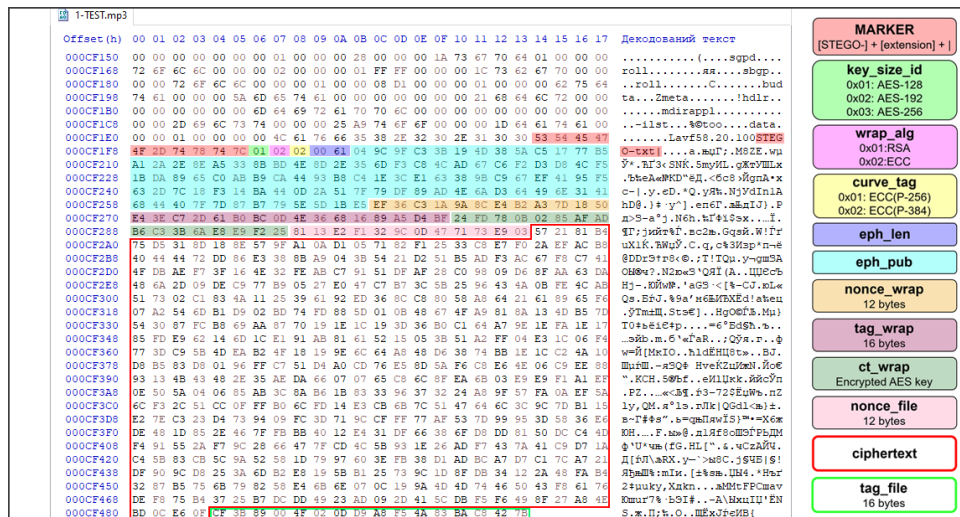


Figure 4: The structure of the created steganographic container.

or ECC – corresponding to the possible values 0x01 and 0x02. In this case, an elliptic curve-based asymmetric algorithm was selected for key encapsulation, so the tag is 0x02.

When using algorithms based on elliptic curves, an additional tag is recorded – “curve tag”, which specifies the particular curve (0x01 for P-256; 0x02 for P-384). This ensures that the parameters are consistent between the sender and the recipient. In this case, ECC-384 was selected.

Next comes service information regarding the encapsulated AES key. This depends on the chosen asymmetric algorithm. When using ECC, the service tag “eph len” is embedded, which represents the specified length of the sender’s ephemeral public key in big-endian format and consists of two bytes. Since ECC-384 was used when forming the steganographic container, the defined length of the ephemeral public key is 97 bytes. This key is formed according to the expression: 0x04 || X || Y, where X and Y are the coordinates of the point’s location, for which 48 bytes are allocated each, and the first byte of the ephemeral public key in the “eph pub” block takes the value 0x04, which represents the “Uncompressed” marker in accordance with the ANSI X9.62 standard. Thus, the “eph len” tag is displayed in a hex editor as 0x61, but if ECC-256 had been selected for encapsulating the AES key, the tag would be displayed as 0x41, as 32 bytes are allocated for the coordinates.

Next, the steganographic container contains the generated “nonce wrap” value (12 bytes), which is used when applying the AES-GCM algorithm to encrypt the AES key prior to encapsulation. Following the “nonce wrap” value, an authentication tag (16 bytes) relating to the encrypted AES key is placed, which is designated as “ct wrap”.

If RSA was selected for encapsulation, only the encrypted AES key “ct wrap” is placed in the steganographic container. Thus, all the described service data form a header, which was used as the AAD together with the “nonce file” value “12 bytes” during the encryption and embedding of the message.

The ciphertext is placed in the steganographic container immediately after the “nonce file”, and the final element is the authentication tag “tag file”, which was generated based on the AAD, “nonce file” and ciphertext.

This proposed approach to crypto-steganographic information protection solves the problem of optimal distribution of hidden data within the container, as the encrypted message containing the service data is placed after a marker embedded following the main content of the container, which does not compromise the quality of the media file after the information has been added to it. Thus, files with virtually any extension can be used both as a container and as a message to be embedded. Furthermore, this approach makes it possible to hide files larger than the container itself. However, it is worth noting that embedding information significantly larger than the container will result in noticeably abnormal file sizes, which may arouse suspicion among third parties not involved in the covert communication.

4. Assessment of the competitiveness of the developed marker-based crypto-steganographic solution

Since the software crypto-steganographic module is designed to conceal information by encrypting it and embedding it after a dedicated marker at the very end of a media file (an empty container), the resulting steganographic container remains indistinguishable from the original container with respect to its primary content. In particular, for image files, the visual appearance of the original and modified files remains identical. Consequently, the mean squared error (MSE) between the original and steganographic images is equal to zero. Since the proposed approach does not modify the media data itself, classical statistical steganalysis methods, including visual analysis, the Chi-square attack, Sample Pair Analysis (SPA), and Regular-Singular Analysis (RS), are generally not applicable for detecting the presence of hidden information. Furthermore, the proposed approach is resistant to file-header-based inspection techniques, as the original file signature and media structure remain unchanged, while the concealed payload is stored in encrypted form after the end of the valid media data. The hidden information is accompanied by service markers that can be modified according to implementation requirements, thereby reducing the effectiveness of signature-based detection methods.

However, similar to many other steganographic approaches, the proposed multi-layer information protection method is sensitive to media file recompression during transmission over open communication channels. Nevertheless, this limitation does not significantly affect practical information exchange scenarios. Most modern messaging and email services automatically process or recode media files when they are inserted directly into a message, which may damage or completely remove embedded hidden information regardless of whether spatial-domain or frequency-domain steganographic techniques are employed. Therefore, one of the key requirements for preserving the integrity of the hidden payload is the correct transmission of steganographic containers. Such containers should be transferred exclusively as file attachments rather than inserted directly into the body of a message.

In summary, the obtained results demonstrate the practical applicability and effectiveness of the proposed crypto-steganographic approach for secure information exchange over open communication channels. The combination of steganographic concealment and cryptographic protection provides an additional layer of security against unauthorized access to confidential information. Even if the existence of hidden data is detected, recovering the concealed information without knowledge of the cryptographic key remains computationally infeasible under the assumed threat model. Moreover, when authenticated encryption algorithms are employed, modification of even a single ciphertext byte results in authentication failure during the decryption process, thereby ensuring the integrity and authenticity of the protected information.

5. Conclusions

The study resulted in a comprehensive analysis of modern cryptographic and steganographic approaches to information security and identified their advantages, limitations, and practical applications in the context of data transmission over open communication channels. It was established that the separate use of cryptographic methods or steganographic methods alone does not provide comprehensive multi-layer information security. Cryptography effectively conceals the content of a message but does not hide the fact of its transmission, whereas steganography conceals the existence of information but does not guarantee its confidentiality if the hidden data are detected. This confirms the feasibility of integrating both approaches within a unified crypto-steganographic system.

Based on the conducted analysis, a novel approach to the design of a crypto-steganographic module for the secure transmission of confidential information was proposed and implemented. The developed solution is based on the use of the AES symmetric encryption algorithm in GCM mode to ensure data confidentiality and integrity, as well as the RSA and ECC asymmetric algorithms for symmetric key encapsulation. An additional level of protection is achieved through a steganographic mechanism based on a dedicated service marker embedded after the main content of the container, allowing encrypted

data to be concealed without modifying the original media content or affecting its normal rendering and playback.

The proposed approach offers several practical advantages. Firstly, the implemented mechanism does not modify the primary content of the container, thereby avoiding the introduction of visual, acoustic, or statistical artefacts that could be detected by conventional steganalysis techniques. Secondly, the system supports a wide range of file formats for both containers and hidden messages, which increases its versatility and applicability. Thirdly, the use of AES-GCM enables the simultaneous implementation of encryption and authentication mechanisms, providing protection against unauthorised modification of transmitted data.

The study also included a detailed analysis of the structure of the generated steganographic container, the organisation of service tags, encryption parameters, and encapsulated key data. It was established that the use of a dedicated service marker simplifies the localisation and extraction of hidden data during the legitimate unpacking process while preserving the flexibility and extensibility of the proposed container structure. Furthermore, the developed architecture supports adaptation to various cryptographic configurations and operating modes.

The practical part of the study confirmed the feasibility of the proposed approach. A software module was implemented in Python using modern cryptographic libraries and multimedia file processing tools. The developed module provides key pair generation, data encryption and decryption, key encapsulation, steganographic container creation, as well as the subsequent extraction and recovery of hidden information. Experimental testing demonstrated the correct functioning of the system and confirmed the effectiveness of the implemented algorithm when working with different container types and file formats.

Thus, the developed crypto-steganographic system provides multi-level information protection by combining the advantages of cryptography and steganography within a single software solution. The proposed approach can be applied in secure communication systems, information and communication infrastructures, software tools for information protection, and scientific research related to cryptography, steganography, and cybersecurity.

Future research should focus on improving information concealment mechanisms through adaptive steganographic techniques, increasing resistance to advanced cryptanalytic and steganalytic methods, supporting secure streaming data transmission, and integrating digital signature mechanisms to provide user authentication and non-repudiation. Additional research may also investigate the resilience of the proposed approach against file-format validation procedures and modern digital forensic analysis techniques.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

- [1] Statista.com, Share of organizations worldwide that have experienced a loss of sensitive information as of February 2024, by country (2026). URL: <https://www.statista.com/statistics/1387392/loss-sensitive-information-organizations-worldwide-by-country/>.
- [2] E. Fernando, D. Agustin, M. Irsan, D. F. Murad, H. Rohayani, D. Sujana, Performance Comparison of Symmetries Encryption Algorithm AES and DES With Raspberry Pi, *International Conference on Sustainable Information Engineering and Technology* (2019) 353–357. doi:10.1109/SIET48054.2019.8986122.
- [3] M. N. Alenezi, H. Alabdulrazzaq, N. Q. Mohammad, Symmetric Encryption Algorithms: Review and Evaluation Study, *International Journal of Communication Networks and Information Security*

- 12 (2) (2022) 256–273. URL: https://www.researchgate.net/publication/349324592_Symmetric_Encryption_Algorithms_Review_and_Evaluation_study.
- [4] S. Pandey, M. Farik, Best Symmetric Key Encryption-A Review, *International Journal of Scientific and Technology Research* 6 (6) (2017) 310–312. URL: https://www.researchgate.net/publication/317913002_Best_Symmetric_Key_Encryption-A_Review.
- [5] D. Mahto, D. K. Yadav, RSA and ECC: A Comparative Analysis, *International Journal of Applied Engineering Research* 12 (19) (2017) 9053–9061. URL: https://www.researchgate.net/publication/322558426_RSA_and_ECC_A_comparative_analysis.
- [6] O. M. Creado, X. Wu, Y. Wang, P. D. Le, Probabilistic Encryption–A Comparative Analysis against RSA and ECC, *Fourth International Conference on Computer Sciences and Convergence Information Technology* (2009) 24–26. doi:10.1109/ICCIT.2009.217.
- [7] M. Suárez-Albela, P. Fraga-Lamas, T. M. Fernández-Caramés, A Practical Evaluation on RSA and ECC-Based Cipher Suites for IoT High-Security Energy-Efficient Fog and Mist Computing Devices, *Sensors* 18 (11) (2018). doi:10.3390/s18113868.
- [8] R. Nadaf, S. Bhairannawar, Design of Hybrid Encryption Algorithm using AES and RSA for key exchange, *Oeil Research Journal* 22 (2024) 75–85. URL: <https://oeilresearch.com/wp-content/uploads/2024/09/8-OEIL-16.pdf>.
- [9] S. K. Akshay, S. P. Vimalbhai, R. A. Rameshbhai, S. M. Ifrah, V. Jadeja, Hybrid AES–RSA Encryption and Decryption for Secure Data Transmission, *International Journal of Research Publication and Reviews* 6 (2025) 1615–1621. doi:10.55248/gengpi.6.1025.3605.
- [10] A. Moruskar, Securing Financial Transactions with Hybrid ECC and AES Encryption Algorithm, 2024. URL: <https://norma.ncirl.ie/8342/1/anujamahadevmoruskar.pdf>.
- [11] P. Selvi, S. Sakthivel, A hybrid ECC-AES encryption framework for secure and efficient cloud-based data protection, *Scientific Reports* 15 (2025). doi:10.1038/s41598-025-01315-5.
- [12] P. Kumar, V. Saxena, K. V. Singh, Analysis of Hybrid Cryptography for Secure Exchange of Information, *International Journal of Computer Applications* 185 (2023) 37–42. doi:10.5120/ijca2023922701.
- [13] I. Almomani, A. Alkhayer, W. El-Shafai, A Crypto-Steganography Approach for Hiding Ransomware within HEVC Streams in Android IoT Devices, *Sensors* 22(6) (2022). doi:10.3390/s22062281.
- [14] O. C. Abikoye, K. S. Adewole, A. J. Oladipupo, Efficient Data Hiding System using Cryptography and Steganography, *International Journal of Applied Information Systems* 4 (2012) 6–11. doi:10.5120/ijais12-450763.
- [15] A. Shifa, M. S. Afgan, M. N. Asghar, M. Fleury, I. Memon, S. Abdullah, Joint Crypto-Stego Scheme for Enhanced Image Protection With Nearest-Centroid Clustering, *IEEE Access* 6 (2018) 16189–16206. doi:10.1109/ACCESS.2018.2815037.
- [16] K. Sethi, B. Sahoo, Shrishti, B. N. Pavan Kumar, K. Dhal, W. Cho, G. P. Joshi, Lightweight DWT Steganography With ECC-ChaCha20 for Secure Medical IoT Systems, *IEEE Access* 13 (2025) 142948–142960. doi:10.1109/ACCESS.2025.3598099.
- [17] A. K. Bermani, High Security Steganography Model Based on DWT, DCT and RSA, *Journal of Engineering and Applied Sciences* 12 (2017) 8875–8881. URL: https://cdnx.uobabylon.edu.iq/research/repository1_publication16106_4_181.pdf.
- [18] K. M. Abdellatif, R. Chotin-Avot, H. Mehrez, Efficient Parallel-Pipelined GHASH for Message Authentication, *International Conference on ReConFigurable Computing and FPGAs* (2012). doi:10.1109/ReConFig.2012.6416742.
- [19] D. A. McGrew, J. Viega, The Galois/Counter Mode of Operation (GCM), 2004. URL: <https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-spec.pdf>.
- [20] Geeksforgeeks.org, RSA Algorithm in Cryptography, 2026. URL: <https://www.geeksforgeeks.org/computer-networks/rsa-algorithm-cryptography/>.
- [21] Datatracker.ietf.org, PKCS 1: RSA Cryptography Specifications Version 2.2, 2016. URL: <https://datatracker.ietf.org/doc/html/rfc8017>.