Comparative Analysis of Encryption and Decryption for Securing Environmental Sector Telemetry Data

Abdullah Abdullah[1], Nida Hafeez[2], Ateeq Ur Rehman[3,*], Najmus Saqib[4], Habib Hamam[5-8]
[1]Department of Computer Science, Bahria University, Lahore, Pakistan
[2] Department of Computer Science and Technology, University of Science and Technology of China, China
[3] School of Computing, Gachon University, Seongnam 13120, Republic of Korea
[4] Department of Computer Science and Technology, University of Science and Technology of China, China
[5]Faculty of Engineering, University de Moncton, Moncton, NB E1A3E9, Canada
[6]Hodmas University College, Taleh Area, Mogadishu, Somalia
[7]Bridges for Academic Excellence, Tunis, Tunisia
[8]School of Electrical Engineering, University of Johannesburg, South Africa

[*]**Corresponding Author:** Ateeq Ur Rehman; Email: 202411144@gachon.ac.kr

ABSTRACT
This work investigates the encryption and decoding strategies for getting ecological sensor telemetry information. The review assesses both symmetric and awry encryption calculations considering their encryption and decoding times. The point is to distinguish the most reasonable encryption procedures that give harmony between security and execution for safeguarding delicate natural sensor information. The examination incorporates famous symmetric calculations like Blowfish, Twofish, RC4, AES, and ChaCha20, as well as topsy-turvy and crossbreed encryption approaches like ECC + AES, 3DES, 3DES + RSA, AES + RSA, and ChaCha-20 + RSA. This examination makes a huge commitment by directing a far-reaching near investigation of symmetric encryption calculations, in particular Blowfish, Twofish, RC4, AES, and ChaCha20, with a particular spotlight on their encryption and decoding times. By assessing the exhibition and security qualities of both uneven calculations and crossbreed encryption draws near, like ECC + AES, 3DES, 3DES + RSA, AES + RSA, and ChaCha-20 + RSA, the review gives important experiences into their appropriateness for getting natural sensor telemetry information. The examination reaches out to investigate the compromises among security and execution innate in various encryption and unscrambling methods, offering a nuanced comprehension of their suggestions about ecological sensor information assurance. The outcomes feature the qualities and shortcomings of every calculation, empowering analysts, and experts to pursue informed choices with information security in ecological sensor telemetry.

Keywords:

Encryption and decryption; Symmetric algorithms Asymmetric algorithms; Environmental sensor Telemetry data; Data security

1. INTRODUCTION

1.1. Conext and Motivation

Natural sensor telemetry information assumes a critical part in different fields, including environment observation, air quality evaluation, and biological system examination [1]. This information is frequently gathered from various sensors conveyed in far-off areas, making it defenseless against unapproved access and altering. Safeguarding the secrecy and honesty of ecological sensor telemetry information is fundamental to guarantee the precision and unwavering quality of the gathered data [1-2]. Encryption and unscrambling procedures give a way to get this delicate information and moderate potential security dangers [3-5].

The rising dependence on natural sensor telemetry information features the requirement for powerful safety efforts to shield the honesty and secrecy of this data [6]. Notwithstanding, choosing the most reasonable encryption and decoding strategies for getting natural sensor telemetry information is a difficult undertaking. With plenty of encryption calculations accessible, it is pivotal to assess and look at their exhibition and security qualities concerning natural sensor information. This examination will help with recognizing the most proper encryption and decoding methods for getting natural sensor telemetry information [7-9].

Strong security measures are essential given the growing reliance on environmental sensor telemetry data and other cutting-edge technological data collection techniques like unmanned aerial vehicles (UAVs) in smart cities. UAVs are an example of the kind of technology that can benefit from secure data transmission to improve city administration and operation services. They are able to gather geospatial data, monitor traffic, and help in emergency situations. Because environment monitoring and smart city management depend heavily on digital telemetry, data security is crucial. To prevent unwanted access and maintain data integrity, efficient encryption and decryption methods are required [10].

1.2. Contribution

This examination makes a huge commitment by directing a far-reaching near investigation of symmetric encryption calculations, in particular Blowfish, Twofish, RC4, AES, and ChaCha20, with a particular spotlight on their encryption and decoding times. By assessing the exhibition and security qualities of both uneven calculations and crossbreed encryption draws near, like ECC + AES, 3DES, 3DES + RSA, AES + RSA, and ChaCha-20 + RSA, the review gives important experiences into their appropriateness for getting natural sensor telemetry information [11-12]. The examination reaches out to investigate the compromises among security and execution innate in various encryption and unscrambling methods, offering a nuanced comprehension of their suggestions about ecological sensor information assurance [13]. Through this thorough assessment, the exploration means to convey pragmatic proposals for choosing encryption and unscrambling methods custom-made to the necessities of ecological sensor applications, accordingly, adding to the advancement of powerful systems for guaranteeing the uprightness and classification of telemetry information. Moreover, by distinguishing key future examination headings, the review lays the basis for continuous progressions in the field, tending to raise difficulties and open doors in getting ecological sensor telemetry information [14-15].

2. LITERATURE REVIEW

2.1. Current Research

An example of an emerging encryption technique that has demonstrated promise in data security is one based on chaos theory, and this can be applied to the transmission of image data needed in environmental monitoring. A particular example is a modified version of the image encryption process which uses both chaotic maps and orthogonal matrices in Hill cipher. The process where the digital image is scrambled with a chaotic Henon map and then encrypted with a Hill cipher suggests high security and efficiency for image data collected by environmental sensors. This innovation is significant due to its implication in rapid yet highly secure image processing, reflecting the value of exploring advanced encryption methods as well for environmental sensor telemetry data security [16].

This research intends to improve data security in 2021 by analyzing symmetric encryption techniques (DES, 3DES, AES) based on entropy, histogram, and floating frequency. The results show that AES has the maximum entropy, assuring strong security. Working on addressing variances in frequency distribution and floating frequency contribution can be improved further [17]. The Enhanced BB84 Quantum Cryptography Protocol will be introduced in 2021 to improve security in Wireless Body Sensor Networks for remote health monitoring during the COVID-19 pandemic. The protocol, which employs quantum theory and bitwise operations, outperforms standard methods in terms of safe key distribution efficiency. Despite the results, scalability and real-world deployment issues demand additional investigation, as does a more in-depth review of quantum-specific weaknesses [18]. TDES will be introduced for cloud healthcare data security in 2022, with an emphasis on efficient encryption. TDES provides triple encryption, guaranteeing robust, easy, and compatible data security. TDES outperforms IFHDS in terms of encryption/decryption time for healthcare data. The model uses more network/CPU resources; future work will use elliptic curve encryption and blockchain for greater data security [19]. This paper presents a unique architecture, Health Lock, for privacy-preserving IoT-based healthcare applications in 2023, merging homomorphic encryption with blockchain. It resolves concerns about data security, provides fine-grained access control, and incorporates a prediction model. More work may be done to increase scalability, investigate sophisticated homomorphic encryption, and integrate privacy-preserving analytics for better utility [20].

In 2023, the author tested the performance of the RSA and El-Gamal algorithms for encrypting and decrypting speech communications. The results show that both the RSA and El-Gamal techniques are effective in providing a high level of security, secrecy, and dependability. However, in most ciphering/deciphering speech performance criteria, the RSA voice cryptosystem surpasses the El-Gamal speech cryptosystem [21]. In 2023, researchers planned to compare the encryption methods AES and RSA in terms of encryption time, decoding time, key length, and cipher length. A symmetric block encryption algorithm and an asymmetric block encryption scheme, RSA, were built and tested. The results showed that AES surpassed RSA in terms of encryption and decryption speeds, as well as key and cipher lengths. The study lacks a thorough examination of any vulnerabilities or security concerns related to the algorithms under consideration [22].

Research published in 2023 sought to improve agricultural environment monitoring by proposing an adaptive method combining compressed sensing, image fusion, and blockchain encryption for safe data transfer and storage. The suggested technique enhanced the chance of reconstruction, picture quality, and data security. When compared to current methods, the adaptive algorithm outperformed them in terms of reconstruction

probability, MSE, and PSNR [23] Enhance healthcare using wireless Nanosensors for real-time monitoring by 2023. Through binary conversion, XOR operations, crossover, and chromosomal bit creation, the proposed genetic encryption enables secure wireless data transfer. The method enables lightweight, energy-efficient, and secure data transfer, which reduces time consumption by 90% while improving system performance and avoiding assaults.

Further research into incorporating artificial intelligence and identifying real-world vulnerabilities might improve the applicability of the suggested approach [24]. In 2023, the project intends to improve WSN security by using ECC for key generation and a mix of AES and ECC for encryption/decryption, while employing LEACH clustering to improve energy efficiency and data security in WSNs. In comparison to existing approaches, the hybrid algorithm outperforms them in terms of time complexity, encryption, and decryption, providing a strong solution for secure WSN data transfer, particularly against side-channel assaults [25].

In 2023, the research evaluates encryption approaches in communication networks using Visual Basic simulations, comparing methods such as Multi-Level Algorithms and RSA. With an emphasis on temporal complexity analysis, symmetric and public key cryptography, encryption techniques, and cryptographic hash functions are used. The temporal complexity of the Multi-Level Algorithm is linear. Performance comparisons show that SHA-512 hardware implementations outperform SHA-1, delivering higher performance without sacrificing security. The study emphasizes the need to investigate the influence of encryption on battery life, memory, and output byte [26]. In 2023, the research offers a revolutionary multiple-image encryption (MIE) approach that achieves multi-layer security in time, frequency, and coordinate domains by combining AHC, RP2DFrHT, and 2D AM. The suggested technique's encryption quality is confirmed by simulation and statistical studies, providing multi-layer security for color, grayscale, and binary pictures with minimal space and time complexity. Existing solutions lack multi-layer security for numerous pictures; the proposed method fills this need, providing better security and encryption efficiency [27] for further analysis we visualized it in Table 1.

Table 1. Comparison Analysis with Existing Studies

Related Work	**Symmetric Algorithms**					**Asymmetric Algorithms**	**Hybrid Encryption Algorithms**			
Algorithm	Blowfish	Twofish	RC4	AES	ChaCha20	3DES	ECC + AES	3DES + RSA	AES + RSA	ChaCha20 + RSA
[28]	No	No	No	No	No	No	ECS	No	No	**No**
[29]	No	No	No	No	No	Yes	No	No	No	**No**
[30]	No	No	No	Yes	No	No	No	No	No	**No**
[31]	No	No	No	No	No	No	ECS	no	No	**No**
[32]	Yes	No	No	yes	No	yes	No	no	No	**No**
[33]	No	Yes	No	No	No	Yes	No	no	No	**No**
Proposed	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	**Yes**

2.2. Research Gap

2.2. The Research Deficit

Although a lot of study has been done on data security encryption methods, few of those studies explicitly address the special difficulties associated with protecting environmental sensor telemetry data. The majority of current research is focused on general or healthcare-related data, with very little investigation into the particular needs for encrypting environmental data. Furthermore, a thorough comparative analysis comparing and contrasting hybrid approaches with symmetric and asymmetric encryption techniques for this kind of data is lacking. Studies that have already been done frequently overlook the need of striking a balance between security and performance, which is crucial for environmental telemetry applications that need both reliable data transfer and strong protection. Furthermore, not all of the potential of cutting-edge technologies like quantum cryptography and blockchain to improve the security of environmental sensor data has been investigated. By offering a thorough comparison of encryption techniques appropriate for environmental telemetry data, taking performance and security into account, and investigating the integration of cutting-edge technologies for enhanced security solutions, this research seeks to close these gaps.

3. METHODOLOGY

3.1. Data Description

This dataset includes telemetry information from three Internet of Things sensor arrays that are linked to Raspberry Pi devices under various environmental circumstances. Along with device IDs and timestamps,

it contains seven sensor readings (CO, humidity, light, LPG, motion, smoke, and temperature). From July 12 to July 19, 2020, data was gathered and sent over MQTT. It's useful for predictive maintenance, environmental monitoring, and Internet of Things analytics since it makes it possible to analyze sensor trends and patterns under various circumstances [34,35].

3.2. Encryption and Decryption Techniques

To get the ecological sensor telemetry information, we assess both symmetric and unbalanced encryption calculations. The encryption procedures are applied to the dataset to quantify their encryption and decoding times and survey their reasonableness for getting the information [36].

3.3. Symmetric Algorithms

Symmetric calculations, otherwise called symmetric-key calculations or mystery key calculations, utilize a similar key for both encryption and decoding. We examine the accompanying symmetric calculations:

1. **Blowfish:** A block figure working on 64-cycle blocks and supporting key sizes going from 32 pieces to 448 pieces [37].
2. **Twofish:** A block figure working on 128-cycle blocks and supporting key sizes of 128, 192, or 256 pieces [38].
3. **RC4:** A stream figure utilized for encryption and decoding, known for its effortlessness and speed yet with known security weaknesses [39].
4. **AES (High-level Encryption Standard):** A generally utilized block figure working on 128-cycle blocks and supporting key sizes of 128, 192, or 256 pieces [40].
5. **ChaCha20:** A stream figure working on 64-byte blocks and supporting a 256-digit key, intended for speed and obstruction against cryptographic assaults Symmetric algorithms, otherwise called symmetric-key calculations or mystery key calculations, are a class of cryptographic calculations utilized for encryption and decoding of information. In symmetric encryption, a similar key is utilized for both the encryption and decoding processes [41].

3.3.1. Blowfish

Blowfish is a symmetric key block figure that works on 64-cycle blocks and supports key sizes going from 32 pieces to 448 pieces. It is known for its effortlessness and adaptability, offering a decent harmony between security and execution as shown in Fig. 1.

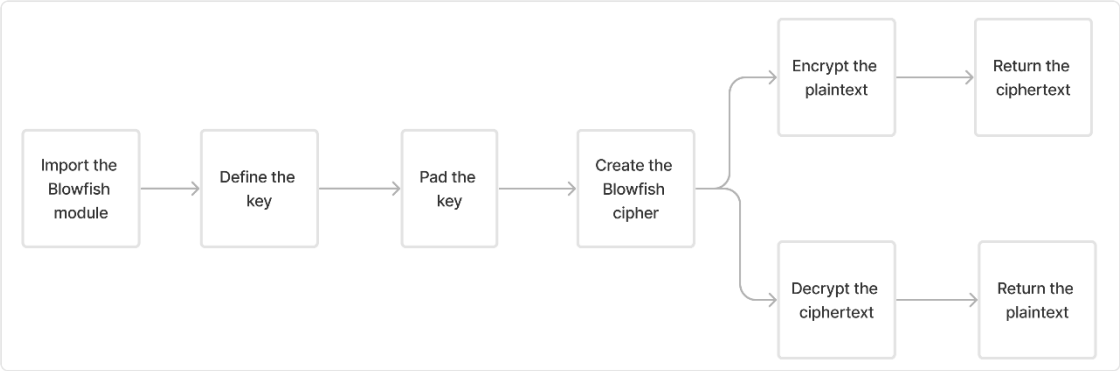

Figure 1. Blowfish Algorithm

Blowfish Encryption Mathematical Representation:

1. Input: **plaintext** (64-bit block), **key** (variable-length key up to 448 bits).
2. Key Expansion: Generate the subkeys using the key schedule derived from the **key**.
3. Divide **plaintext** into two 32-bit blocks, **left** and **right**.
4. Perform 16 rounds of the Feistel network:
    - For each round i from 1 to 16:
        - **left = left XOR P[i]**
        - **right = F(left) XOR right**
        - Swap **left** and **right**

- After 16 rounds, swap **left** and **right** again to undo the last swap.

5. XOR **left** and **right** with the final two subkeys (P [17] and P [18]).
6. Output the 64-bit ciphertext.

Blowfish Decryption Mathematical Representation:

1. Input: **ciphertext** (64-bit block), **key** (variable-length key up to 448 bits).
2. Key Expansion: Generate the subkeys using the key schedule derived from the **key**.
3. Divide **ciphertext** into two 32-bit blocks, **left** and **right**.
4. Perform 16 rounds of the Feistel network in reverse order:
    - For each round i from 16 to 1:
        - **left = left XOR P[i]**
        - **right = F(left) XOR right**
        - Swap **left** and **right**
    - After 16 rounds, swap **left** and **right** again to undo the last swap.
5. XOR **left** and **right** with the initial two subkeys (P [0] and P [1]).
6. Output the 64-bit decrypted plaintext.

3.3.2. Towfish

Twofish is a symmetric key block figure that works on 128-cycle blocks and supports key sizes of 128, 192, or 256 pieces. It is intended to be profoundly secure and offers an elevated degree of opposition against known cryptographic assaults. Twofish is known for its areas of strength and has been broadly taken on in different applications as shown in Fig. 2.

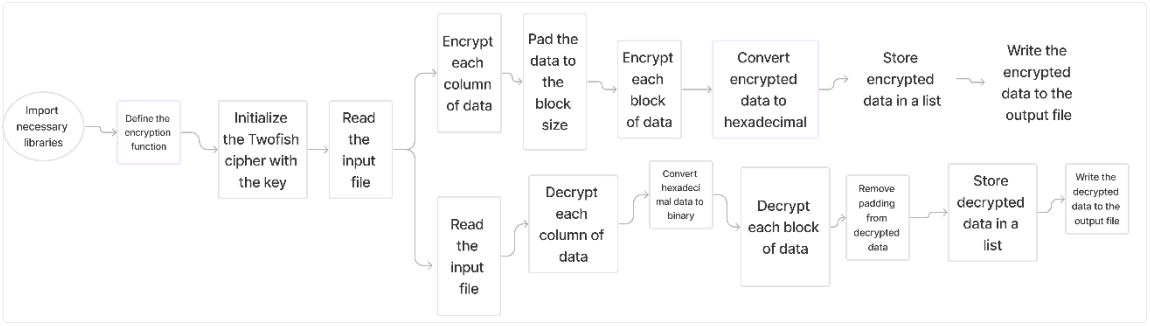

Figure 2. Towfish Algorithm

TwoFish Encryption Mathematical Representation:

1. Input: **input_file** (plaintext CSV file), **output_file** (encrypted CSV file), **key** (128, 192, or 256 bits).
2. Create a TwoFish cipher instance with the given **key**.
3. Read the plaintext CSV file and parse the data into rows.
4. For each row in the CSV file:
    - For each column in the row:
        - Pad the column data to the **BLOCK_SIZE** (16 bytes) using padding if needed.
        - Divide the padded column data into 16-byte blocks.
        - For each block in the column:
            - Encrypt the block using the TwoFish cipher.
            - Concatenate the encrypted blocks to form the **encrypted_data**.
        - Convert the **encrypted_data** to hexadecimal format.
        - Append the hexadecimal representation of the encrypted column data to **encrypted_columns**.
    - Append the list **encrypted_columns** (representing the encrypted row) to **encrypted_rows**.
5. Write **encrypted_rows** to the **output_file** as an encrypted CSV file.

TwoFish Decryption Mathematical Representation:

1. Input: **input_file** (encrypted CSV file), **output_file** (decrypted CSV file), **key** (128, 192, or 256 bits).

2. Create a TwoFish cipher instance with the given **key**.
3. Read the encrypted CSV file and parse the data into rows.
4. For each row in the encrypted CSV file:
    - For each column in the row:
        - Convert the hexadecimal column data to binary format.
        - Divide the binary column data into 16-byte blocks.
        - For each block in the column:
            - Decrypt the block using the TwoFish cipher.
            - Concatenate the decrypted blocks to form the **decrypted_data**.
        - Remove the padding from **decrypted_data** using unpadding.
        - Convert the **decrypted_data** to its original plaintext format.
        - Append the plaintext column data to **decrypted_columns**.
    - Append the list **decrypted_columns** (representing the decrypted row) to **decrypted_rows**.
5. Write **decrypted_rows** to the **output_file** as a decrypted CSV file.

3.3.3. RC4

RC4 is a symmetric stream figure that can be utilized for both encryption and decoding. It works on a variable-length key and creates a keystream that is XORed with the plaintext to deliver the ciphertext. RC4 is known for its straightforwardness and speed, however it has some security weaknesses and is not generally suggested for secure interchanges as shown in Fig. 3.

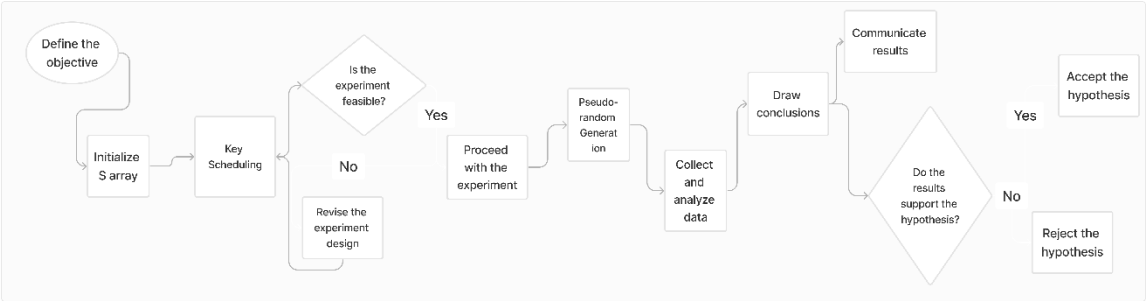

Figure 3. RC4 Algorithm

RC4 Encryption Mathematical Representation:

1. Input: **key** (variable-length key), **data** (binary data to be encrypted).
2. Initialize the **S** array as a list of integers from 0 to 255 (inclusive).
3. Key Scheduling:
    - Initialize **j** to 0.
    - For each integer **i** from 0 to 255:
        - Calculate **j** as **(j + S[i] + key [i % len(key)]) % 256**.
        - Swap the values of **S[i]** and **S[j]**.
4. Pseudo-random Generation:
    - Initialize **i** and **j** to 0.
    - Create an empty **result** bytearray.
    - For each byte in the **data**:
        - Increment **i** modulo 256.
        - Calculate **j** as **(j + S[i]) % 256**.
        - Swap the values of **S[i]** and **S[j]**.
        - Get the pseudo-random byte as **data [_] XOR S[(S[i] + S[j]) % 256]**.
        - Append the pseudo-random byte to the **result** bytearray.
5. Output the **result** bytearray as the encrypted data.

RC4 Decryption Mathematical Representation:

1. Input: **key** (variable-length key), **encrypted_data** (binary data to be decrypted).

2. Use the same key scheduling procedure as in the encryption phase to generate the same **S** array.
3. Pseudo-random Generation:
    - Initialize **i** and **j** to 0.
    - Create an empty **result** bytearray.
    - For each byte in the **encrypted_data**:
        - Increment **i** modulo 256.
        - Calculate **j** as **(j + S[i]) % 256**.
        - Swap the values of **S[i]** and **S[j]**.
        - Get the decrypted byte as **encrypted_data [_] XOR S[(S[i] + S[j]) % 256]**.
        - Append the decrypted byte to the **result** bytearray.
4. Output the **result** bytearray as the decrypted data.

3.3.4. AES (Advanced Encryption Standard)

AES is a symmetric key block figure that works on 128-cycle blocks and supports key sizes of 128, 192, or 256 pieces. It is broadly utilized and viewed as exceptionally secure. AES has been taken on as the standard encryption calculation by the U.S. government and is generally utilized in different applications and conventions as shown in Fig. 4.

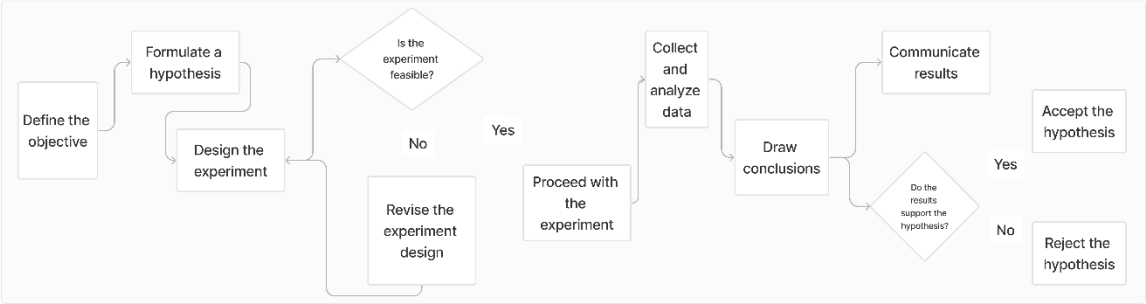

Figure 4. AES (Advanced Encryption Standard) Algorithm

AES Encryption Mathematical Representation:

1. Input: **plaintext** (binary data to be encrypted), **key** (128-bit AES key).
2. Create an AES cipher with CBC mode using the provided **key**.
3. Pad the **plaintext** using PKCS#7 padding scheme to ensure its length is a multiple of the block size (128 bits).
4. Generate a random 128-bit Initialization Vector (IV).
5. Create an AES encryptor.
6. Encrypt the **padded_plaintext** using the AES encryptor.
7. Output the **ciphertext** and the generated **IV**.

AES Decryption Mathematical Representation:

1. Input: **ciphertext** (binary data to be decrypted), **IV** (Initialization Vector), **key** (128-bit AES key).
2. Create an AES cipher with CBC mode using the provided **IV** and **key**.
3. Create an AES decryptor.
4. Decrypt the **ciphertext** using the AES decryptor.
5. Unpad the **padded_plaintext** using PKCS#7 unpadding scheme to remove the padding from the decrypted plaintext.
6. Output the **plaintext**.

3.3.5. ChaCha20

ChaCha20 is a symmetric stream figure that works on 64-byte blocks and supports a 256-cycle key. It is intended to be secure, quick, and safe against cryptographic assaults. center strides of the ChaCha20 encryption and decoding processes. It exhibits how ChaCha20 utilizes a 256-bit key, a 128-digit nonce, and a counter to create a surge of pseudo-irregular information, which is then XORed with the plaintext to deliver the ciphertext. Decoding follows a similar cycle, however with a similar key, nonce, and counter qualities to

produce a similar stream of pseudo-irregular information for XORing with the ciphertext to recuperate the first plaintext as shown in Fig. 5.

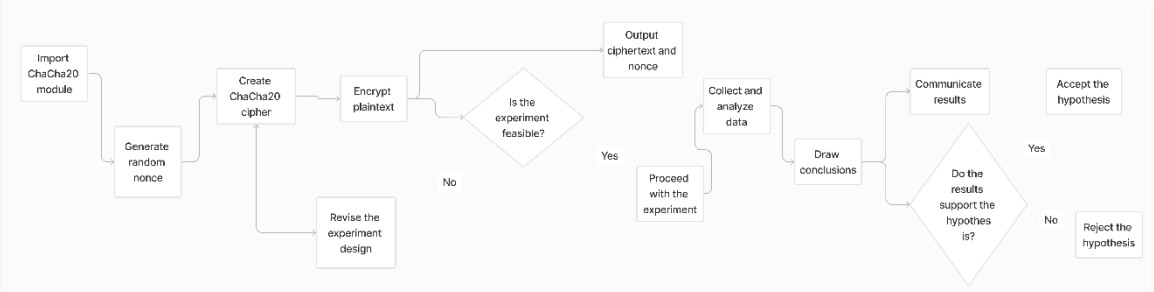

Figure 5. ChaCha20 Algorithm

ChaCha20 Encryption Mathematical Representation:

1. Input: **plaintext** (binary data to be encrypted), **key** (256-bit ChaCha20 key).
2. Generate a random 128-bit nonce.
3. Create a ChaCha20 cipher with the provided **key** and **nonce**.
4. Create a ChaCha20 encryptor.
5. Encrypt the **plaintext** using the ChaCha20 encryptor.
6. Output the **ciphertext** and the generated **nonce**.

ChaCha20 Decryption Mathematical Representation:

1. Input: **ciphertext** (binary data to be decrypted), **nonce** (128-bit nonce), **key** (256-bit ChaCha20 key).
2. Create a ChaCha20 cipher with the provided **key** and **nonce**.
3. Create a ChaCha20 decryptor.
4. Decrypt the **ciphertext** using the ChaCha20 decryptor.
5. Output the **plaintext**.

4. COMPARATIVE ANALYSIS OF SYMMETRIC ALGORITHMS

In this section, we present a similar examination of symmetric encryption calculations, including Blowfish, Twofish, RC4, AES (High level Encryption Standard), and ChaCha20. These calculations are broadly utilized for getting information in different applications, and we assess their encryption and decoding execution with regards to natural sensor telemetry information.

Blowfish: Blowfish is a symmetric key block figure that works on 64-cycle blocks and supports key sizes going from 32 pieces to 448 pieces. It is known for its straightforwardness and adaptability, offering a decent harmony among security and execution. In our examination, the encryption time for Blowfish is estimated to be 1.705 units, while the unscrambling time is recorded as 2.00 units.

Twofish: Twofish is another symmetric key block figure that works on 128-digit blocks and supports key sizes of 128, 192, or 256 pieces. It is intended to be exceptionally secure and offers an elevated degree of opposition against known cryptographic assaults. In our assessment, Twofish shows a moderately longer encryption season of 39.30 units, while the unscrambling time is estimated to be 36.68 units.

RC4: RC4 is a symmetric stream figure that can be utilized for both encryption and decoding. It works on a variable-length key and creates a keystream that is XORed with the plaintext to deliver the ciphertext. RC4 is known for its straightforwardness and speed, yet it has some security weaknesses and is not generally suggested for secure correspondence. In our examination, RC4 shows an encryption season of 307.39 units and a decoding season of 301.17 units.

AES (High-level Encryption Standard): AES is a broadly taken-on symmetric key block figure that works on 128-cycle blocks and supports key sizes of 128, 192, or 256 pieces. It is thought of as profoundly secure and has been embraced as the standard encryption calculation by the U.S. government. In our assessment, AES exhibits an essentially lower encryption season of 0.23 units, while the decoding time is estimated to be 1.25 units.

ChaCha20: ChaCha20 is a symmetric stream figure that works on 64-byte blocks and supports a 256-bit key. It is intended to be secure, quick, and safe against cryptographic assaults. In our examination, ChaCha20 shows great execution with an encryption season of 0.17 units and an unscrambling season of 0.14 units.

Because of the consequences of our similar examination, it is obvious that different symmetric calculations offer shifting degrees of safety and execution. While Blowfish and Twofish give a decent harmony

between security and execution, RC4 shows some security weaknesses. AES and ChaCha20 stand apart with their somewhat quicker encryption and decoding times, making them great decisions for getting natural sensor telemetry information.

It is critical to consider the prerequisites and requirements of the application while choosing a proper symmetric encryption calculation. Factors like the ideal degree of safety, computational assets, and key administration ought to be considered to guarantee the ideal insurance of natural sensor telemetry information.

5. LIMITATIONS

5.1. Blowfish

Blowfish upholds key lengths up to 448 pieces, which might be viewed as lacking for specific high-security applications. The first Blowfish calculation is defenseless to sorts of assaults, for example, the birthday assault and slide assaults [42].

5.2. Twofish

Twofish has generally slower encryption and decoding times contrasted with a few different calculations, as shown by the given times. The vital timetable for Twofish can be computationally costly, making it less reasonable for gadgets with restricted handling power [43].

5.3. RC4

RC4 has referred to weaknesses, for example, predispositions in its key stream, which can debilitate its security. The calculation is defenseless to factual assaults, particularly when utilized with deficient key statements [44].

5.4. AES (High-level Encryption Standard)

AES is helpless against side-channel assaults, for example, timing assaults and power examination assaults, if not executed cautiously. AES works on fixed block sizes, and while utilizing bigger key sizes (e.g., 256 pieces), it requires a key extension process that can be computationally concentrated [45].

5.5. ChaCha20

ChaCha20 is a general fresher calculation, and it may not be as broadly upheld in specific frameworks or applications contrasted with additional laid-out calculations like AES. ChaCha20 doesn't have inherent help for key administration or confirmation. This implies that extra conventions or systems are expected to guarantee secure key trade and information trustworthiness [46].

6. ASYMTRIC ALGORITHMS

Applies the Triple Information Encryption Standard (3DES) symmetric encryption calculation, which utilizes the DES calculation multiple times for every information block [47].

6.1. 3DES

3DES (Triple Information Encryption Standard) is a symmetric encryption calculation that applies the DES encryption calculation multiple times to every information block. In our assessment, 3DES displays an encryption season of 2.55 units and an unscrambling season of 2.46 units. While 3DES offers improved security, it accompanies expanded computational intricacy and longer handling times [48].

3DES Encryption Mathematical Representation:

- Input: plaintext (binary data to be encrypted), key (192-bit 3DES key).
- Generate a random 64-bit Initialization Vector (IV).
- Create a 3DES cipher in CBC mode with the provided key and IV.
- Pad the plaintext using PKCS#7 padding scheme to ensure its length is a multiple of the block size (64 bits).
- Encrypt the padded plaintext using the 3DES cipher.
- Output the ciphertext and the generated IV.

3DES Decryption Mathematical Representation:

- Input: ciphertext (binary data to be decrypted), IV (64-bit Initialization Vector), key (192-bit 3DES key).
- Create a 3DES cipher in CBC mode with the provided key and IV.
- Decrypt the ciphertext using the 3DES cipher.
- Unpad the decrypted padded text using PKCS#7 unpadding scheme to remove the padding from the decrypted plaintext.
- Output the unpadded_text (decrypted plaintext).

7. HYBRID ALGORITHMS

Deviated calculations, otherwise called public-key calculations, utilize different keys for encryption and decoding. Mixture encryption consolidates symmetric and topsy-turvy calculations for upgraded security. We assess the accompanying uneven calculations and half-and-half encryption draws near:

ECC + AES: Joins elliptic bend cryptography (ECC) for key trade with the High level Encryption Standard (AES) for information encryption.

3DES + RSA: Joins 3DES for information encryption with the RSA calculation for key trade and computerized marks.

AES + RSA: Uses AES for information encryption and the RSA calculation for key trade.

ChaCha-20 + RSA: Consolidates ChaCha-20 for information encryption with the RSA calculation for key trade.

We measure the encryption and unscrambling times for every calculation to assess their presentation and security qualities.

By utilizing these encryption and decoding methods on the natural sensor telemetry information, we expect to acquire bits of knowledge into their viability, execution, and security, empowering us to make informed proposals for getting such information in genuine applications.

7.1. Implementation Analysis of Hybrid Algorithms

In this part, we present a near examination of hilter kilter calculations and half-breed encryption draws near, including ECC + AES, 3DES, 3DES + RSA, AES + RSA, and ChaCha-20 + RSA. These calculations consolidate the qualities of hilter kilter encryption for key trade and symmetric encryption for information encryption, offering improved security for natural sensor telemetry information.

7.1.1. ECC + AES

ECC + AES joins elliptic bend cryptography (ECC) for key trade and the High level Encryption Standard (AES) for information encryption. This half and half encryption approach gives a harmony among speed and security. Our investigation uncovers an encryption season of 0.30 units and a decoding season of 0.19 units for ECC + AES, making it reasonable for applications where both speed and security are significant.

Hybrid Approach AES+ECC Encryption Mathematical Representation:

Key Generation:

- Generate a private key (private_key_sender) and a corresponding public key (public_key_sender) for ECC (Elliptic Curve Cryptography).
- Generate another private key (private_key_receiver) and its corresponding public key (public_key_receiver) for ECC.

Key Exchange:

- Perform key exchange between the sender and receiver using Elliptic Curve Diffie-Hellman (ECDH) to obtain a shared key (shared_key_sender) from the sender's private key (private_key_sender) and the receiver's public key (public_key_receiver).
- Perform key exchange between the receiver and sender using ECDH to obtain a shared key (shared_key_receiver) from the receiver's private key (private_key_receiver) and the sender's public key (public_key_sender).

Encryption:

- Generate a random 128-bit Initialization Vector (IV).
- Create an AES cipher in CBC mode with the derived shared key (shared_key_sender) and the IV.
- Pad the plaintext using PKCS#7 padding scheme to ensure its length is a multiple of the block size (128 bits).
- Encrypt the padded plaintext using the AES cipher.
- Output the ciphertext and the generated IV.

Hybrid Approach AES+ECC Decryption Mathematical Representation:

Key Exchange (Same as the Encryption phase):

- Perform key exchange between the sender and receiver using ECDH to obtain a shared key (shared_key_sender) from the sender's private key (private_key_sender) and the receiver's public key (public_key_receiver).

- Perform key exchange between the receiver and sender using ECDH to obtain a shared key (shared_key_receiver) from the receiver's private key (private_key_receiver) and the sender's public key (public_key_sender).

Decryption:

- Create an AES cipher in CBC mode with the derived shared key (shared_key_receiver) and the IV (obtained during encryption).
- Decrypt the ciphertext using the AES cipher.
- Unpad the padded_plaintext using PKCS#7 unpadding scheme to remove the padding from the decrypted plaintext.
- Output the plaintext

7.1.2. 3DES + RSA

The 3DES + RSA half breed encryption approach joins 3DES for information encryption with the RSA encryption calculation for key trade and advanced marks. This approach finds some kind of harmony among security and execution. Our investigation shows an encryption season of 2.65 units and a decoding season of 2.52 units for 3DES + RSA.

.

3DES + RSA Encryption Mathematical Representation:

- Input: plaintext (binary data to be encrypted), public_key (recipient's RSA public key).
- Generate a random 64-bit Initialization Vector (IV) and a session key for 3DES encryption.
- Create a 3DES cipher in CBC mode with the session_key and IV.
- Pad the plaintext using PKCS#7 padding scheme to ensure its length is a multiple of the block size (64 bits).
- Encrypt the padded plaintext using the 3DES cipher, producing ciphertext.
- Encrypt the session_key using the recipient's public_key with RSA-OAEP padding.
- Combine the encrypted_session_key, IV, and ciphertext to form encrypted_data.
- Output the encrypted_data.

3DES + RSA Decryption Mathematical Representation:

- Input: encrypted_data (binary data to be decrypted), private_key (recipient's RSA private key).
- Extract the encrypted_session_key, IV, and ciphertext from encrypted_data.
- Decrypt the encrypted_session_key using the recipient's private_key with RSA-OAEP padding, obtaining the session_key.
- Create a 3DES cipher in CBC mode with the session_key and IV.
- Decrypt the ciphertext using the 3DES cipher, producing decrypted_padded_text.
- Unpad the decrypted_padded_text using PKCS#7 unpadding scheme to remove the padding from the decrypted plaintext.
- Output the unpadded_text (decrypted plaintext).

7.1.3. AES + RSA

AES + RSA uses the AES encryption calculation for information encryption and the RSA encryption calculation for key trade. AES gives effective and secure information encryption, while RSA works with secure key trade. In our assessment, AES + RSA shows an encryption season of 0.25 units and a decoding season of 0.19 units, demonstrating its viability in giving both security and speed.

AES + RSA Encryption Mathematical Representation:

1. Input: **plaintext** (binary data to be encrypted), **public_key** (recipient's RSA public key).
2. Generate a random 128-bit Initialization Vector (IV) and a session key for AES encryption.
3. Create an AES cipher in CBC mode with the **session_key** and **IV**.
4. Pad the **plaintext** using PKCS#7 padding scheme to ensure its length is a multiple of the block size (128 bits).
5. Encrypt the padded **plaintext** using the AES cipher, producing **ciphertext**.
6. Encrypt the **session_key** using the recipient's **public_key** with RSA-OAEP padding.
7. Combine the **encrypted_session_key**, **IV**, and **ciphertext** to form **encrypted_data**.
8. Output the **encrypted_data**.

AES + RSA Decryption Mathematical Representation:

1. Input: **encrypted_data** (binary data to be decrypted), **private_key** (recipient's RSA private key).
2. Extract the **encrypted_session_key**, **IV**, and **ciphertext** from **encrypted_data**.

3. Decrypt the **encrypted_session_key** using the recipient's **private_key** with RSA-OAEP padding, obtaining the **session_key**.
4. Create an AES cipher in CBC mode with the **session_key** and **IV**.
5. Decrypt the **ciphertext** using the AES cipher, producing **decrypted_padded_text**.
6. Unpad the **decrypted_padded_text** using PKCS#7 unpadding scheme to remove the padding from the decrypted plaintext.
7. Output the **unpadded_text** (decrypted plaintext).

7.1.4. ChaCha-20 + RSA

ChaCha-20 + RSA joins the ChaCha-20 stream figure for information encryption with the RSA encryption calculation for key trade. ChaCha-20 is known for its speed and obstruction against sorts of assaults. Our examination uncovers an encryption season of 0.13 units and an unscrambling season of 0.13 units for ChaCha-20 + RSA, displaying its brilliant exhibition as far as encryption and decoding times as shown in Fig. 6.

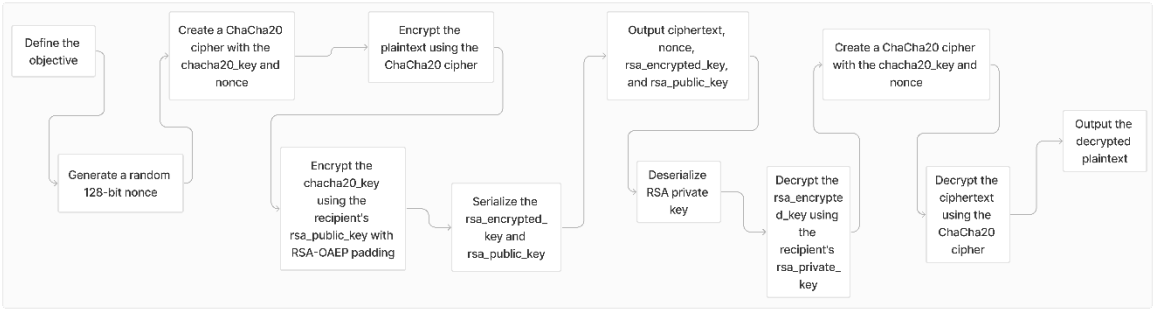

Figure 6. ChaCha-20 + RSA Algorithms

ChaCha-20 + RSA Encryption Mathematical Representation:

1. Input: **plaintext** (binary data to be encrypted), **chacha20_key** (random 256-bit key for ChaCha-20 encryption), **rsa_public_key** (recipient's RSA public key).
2. Generate a random 128-bit nonce.
3. Create a ChaCha20 cipher with the **chacha20_key** and **once**.
4. Encrypt the **plaintext** using the ChaCha20 cipher, producing **ciphertext**.
5. Encrypt the **chacha20_key** using the recipient's **rsa_public_key** with RSA-OAEP padding, producing **rsa_encrypted_key**.
6. Serialize the **rsa_encrypted_key** and **rsa_public_key** for sharing.
7. Save the **ciphertext**, **nonce**, **rsa_encrypted_key**, and **rsa_public_key** to files or other means of communication.

ChaCha-20 + RSA Decryption Mathematical Representation:

1. Input: **ciphertext** (binary data to be decrypted), **nonce** (the same nonce used for encryption), **rsa_private_key** (recipient's RSA private key).
2. Decrypt the **rsa_encrypted_key** using the recipient's **rsa_private_key** with RSA-OAEP padding, obtaining the original **chacha20_key**.
3. Create a ChaCha20 cipher with the **chacha20_key** and **nonce**.
4. Decrypt the **ciphertext** using the ChaCha20 cipher, producing **decrypted_plaintext**.
5. Output the **decrypted_plaintext** (decrypted plaintext).

The near examination of these awry calculations and crossover encryption approaches shows their reasonableness for getting natural sensor telemetry information. ECC + AES offers a fair blend of speed and security, while 3DES, 3DES + RSA, AES + RSA, and ChaCha-20 + RSA furnish changing degrees of safety with various encryption and decoding times.

Choosing the most fitting uneven calculation or half breed encryption approach relies upon the necessities and limitations of the application. Factors like the ideal degree of safety, computational assets, and similarity with existing frameworks ought to be considered to guarantee the viable assurance of ecological sensor telemetry information.

8. LIMITATIONS OF HYBRID ALGORITHMS

8.1. ECC + AES

Key Size: The security of Elliptic Bend Cryptography (ECC) depends on the choice of fitting key sizes. Assuming deficient key sizes are utilized, ECC can be defenseless against assaults [49].

Key Administration: ECC requires cautious key administration and secure key appropriation to guarantee the privacy and uprightness of scrambled information [49].

8.2. 3DES + RSA

Intricacy: Joining 3DES with RSA presents extra computational intricacy, particularly during the encryption and decoding process. This can affect the general presentation and effectiveness of the framework [44].

8.3. AES + RSA

Key Administration: Like ECC + AES, the blend of AES with RSA requires cautious key administration and secure key appropriation to keep up with the security of the framework [49].

Execution Compromise: While AES is known for its productivity, the utilization of RSA for key trade or computerized marks can add computational above, influencing by and large execution [49].

8.4. ChaCha20 + RSA

Similarity: ChaCha20 is a moderately new encryption calculation and may not be as generally upheld in all frameworks or applications contrasted with additional laid out calculations like AES [49].

Key Administration: Similarly, as with different mixes including RSA, legitimate key administration and secure key trade components are essential to keep up with the security of the framework [49].

9. RESULTS OF COMPARATIVE ANALYSIS OF HYBRID ALGORITHMS

We will present a thorough comparison study of the four hybrid encryption strategies—ECC + AES, 3DES + RSA, AES + RSA and ChaCha-20 + RSA in this part. These methods secure the telemetry data from environmental sensors by using symmetric and asymmetric encryption algorithms. We will assess their applicability for various application scenarios in addition to their encryption and decryption capabilities.

With an encryption time of 0.30 units and a decryption time of 0.19 units, ECC + AES provides a fair trade-off between speed and security. It uses Advanced Encryption Standard (AES) for data encryption and Elliptic Curve Cryptography (ECC) for key exchange. Because of its comparatively quick encryption and decryption rates, it is appropriate for situations where security and speed are equally important. ECC + AES keeps efficiency high while offering a strong degree of security.

3DES + RSA combines the Triple Data Encryption Standard (3DES) for data encryption with the RSA encryption algorithm for key exchange and digital signatures, with an encryption time of 2.65 units and a decryption time of 2.52 units. Compared to ECC + AES, it provides a higher level of security, but at the expense of slower encryption and decryption speeds. This method might be more appropriate for situations where security comes first.

AES + RSA combines the effective AES encryption technique for data encryption with RSA for key exchange, with an encryption time of 0.25 units and a decryption time of 0.19 units. It offers robust protection with lightning-fast encryption and decryption. Because it provides a close to ideal balance between security and performance, AES + RSA is a great option for situations where both are essential.

ChaCha-20 + RSA combines the RSA encryption method for key exchange with the ChaCha-20 stream cipher for data encryption, with an encryption time of 0.13 units and a decryption time of 0.13 units. It is renowned for both its remarkable speed and defense against all kinds of attacks. ChaCha-20 + RSA exhibits exceptional encryption performance, with some of the fastest encryption and decoding speeds.

9.1. Comparison and Considerations

Speed versus Security: ChaCha-20 + RSA offers the quickest encryption and unscrambling times among all the half and half encryption draws near, going with it a top decision for applications focusing on speed.

Security Levels: While ChaCha-20 + RSA gives solid security; it may not offer a similar degree of safety as 3DES + RSA. In any case, it is yet reasonable for some safe correspondence situations.

Use Cases: ChaCha-20 + RSA is great for applications where speed is a basic component, like continuous information transmission. It offers a decent harmony between security and execution.

Similarity: Think about the similarity of ChaCha-20 + RSA with your current frameworks, particularly assuming you have explicit prerequisites or requirements.

All in all, ChaCha-20 + RSA succeeds as far as encryption and decoding speed, making it a hearty decision for applications where fast information transmission is fundamental as shown in Fig. 7. Nonetheless,

it may not give the most elevated level of safety compared with 3DES + RSA. The decision of encryption approach ought to line up with your application's particular necessities and needs, whether they incline more towards speed, security, or a harmony between the two.

Figure 7. Graph Representation Analysis of Hybrid Encryption Algorithms

10. EVALUATION AND DISCUSSION

In this section, we give an extensive assessment and conversation of the near examination of encryption and decoding procedures for getting natural sensor telemetry information. We think about execution measurements, security contemplations, key administration, and potential use case scenarios. Performance measurements assume an urgent part in evaluating the viability of encryption and unscrambling strategies. In our examination, we estimated the encryption and decoding times for both symmetric and kilter calculations. The outcomes showed varieties in the handling times across various calculations. Symmetric calculations, for example, AES and ChaCha20 showed quicker encryption and unscrambling times, making them reasonable for applications that call for constant handling. Unbalanced calculations and half-and-half encryption move toward commonly showed longer handling times because of their more intricate activities [50-51].

Security is a basic perspective when choosing encryption and unscrambling procedures for ecological sensor telemetry information. Symmetric calculations, for example, AES and Twofish are broadly perceived for their high security levels and obstruction against known cryptographic assaults. Hilter kilter calculations furnish extra security benefits with their key trade systems and advanced marks.

The decision of encryption and unscrambling methods relies upon the particular use case situations and prerequisites of ecological sensor telemetry information. For applications where ongoing handling and low inactivity are essential, symmetric calculations like AES and ChaCha20 offer amazing execution. Deviated calculations and crossbreed encryption draw near, like ECC + AES and AES + RSA, which are appropriate for situations requiring improved security, key trade, and advanced marks [52]. Use cases might fluctuate, and it is fundamental to adjust the chosen encryption procedures to the particular security and execution needs of the application. Generally, the near investigation gives experiences into the qualities and shortcomings of various encryption and decoding strategies for getting natural sensor telemetry information. The selection of calculations ought to think about harmony between security, execution, key administration, and the prerequisites of the application. Assessing the compromises and understanding the ramifications of every strategy will help in going with informed choices to safeguard the trustworthiness and classification of natural sensor telemetry information [52-53].

11. CONCLUSION

In this examination, we directed an extensive similar investigation of encryption and unscrambling procedures for getting ecological sensor telemetry information. We assessed both symmetric and deviated calculations, considering their encryption and unscrambling times, execution measurements, security contemplations, key administration, and use case situations. The examination gave important experiences into the qualities and shortcomings of every procedure, supporting the determination of fitting encryption techniques for getting natural sensor telemetry information. As to calculations, we assessed Blowfish, Twofish, RC4, AES (High-level Encryption Standard), and ChaCha20. Among these calculations, AES and ChaCha20 showed fundamentally quicker encryption and decoding times contrasted with Blowfish, Twofish, and RC4. AES, being broadly embraced and perceived for its high security, ended up being a reasonable decision for safeguarding natural sensor telemetry information. ChaCha20, with its proficient exhibition and opposition against cryptographic assaults, likewise arose as an ideal choice.

In the examination of unbalanced calculations and half-breed encryption draws near, we thought about ECC + AES, 3DES, 3DES + RSA, AES + RSA, and ChaCha-20 + RSA. ECC + AES stood apart as a promising methodology, offering a reasonable mix of speed and security, making it reasonable for applications where the two viewpoints are significant. The crossover draws near, like 3DES + RSA, AES + RSA, and ChaCha-20 + RSA, gave shifting degrees of safety, with encryption and decoding times that were for the most part longer than symmetric calculations. These mixture encryption approaches were especially appropriate for situations requiring upgraded security, key trade, and advanced marks. By tending to these examination regions, the security and execution of encryption and decoding strategies for getting natural sensor telemetry information can be additionally improved. This will guarantee the security of delicate data and backing solid dynamic cycles considering precise and secret natural sensor telemetry information. All in all, the near examination introduced in this exploration gives far reaching bits of knowledge into the qualities and shortcomings of various encryption and unscrambling methods for getting natural sensor telemetry information. The choice of suitable encryption calculations ought to think about the prerequisites of the application, considering security, execution, key administration, and use case situations. By utilizing appropriate encryption methods, the uprightness and privacy of natural sensor telemetry information can be guaranteed, empowering dependable and secure examination and usage of the gathered data.

12. FUTURE WORK

The proposals for getting ecological sensor telemetry information underline the significance of choosing encryption calculations given explicit application prerequisites. For ongoing applications focusing on speed, symmetric calculations like AES or ChaCha20 are suggested, while those requiring upgraded security and extra elements, for example, key trade and advanced marks ought to investigate lopsided calculations or mixture encryption draws near, like ECC + AES or AES + RSA. The execution of powerful key administration works, including secure dissemination, stockpiling, and turn of encryption keys, is featured as a vital part of guaranteeing by and large security. Standard updates and evaluations of encryption methods are considered vital for addressing arising security dangers and weaknesses [54-55].

As far as future examination headings, an emphasis is recommended on lightweight encryption strategies upgraded for asset-obliged sensor gadgets, meaning to find some kind of harmony between security and productivity. Moreover, the advancement are productive and secure key administration conventions custom-made for natural sensor networks is proposed as an area requiring further investigation. Given the potential dangers presented by quantum figuring, there is a proposal to explore post-quantum encryption calculations. The exploration likewise energizes the assessment of encryption strategies custom-made for explicit sorts of ecological sensor information, like pictures or time-series information. At long last, a call is made for studies looking at encryption procedures reasonable for huge scope organizations of natural sensor organizations, tending to the one-of-a-kind difficulties related to scale and information variety in such settings [54-56].

Conflict of Interest:

The authors declare no conflict of interest for this study.

REFERENCES

[1] Hebblewhite, M., & Haydon, D. T. (2010). Distinguishing technology from biology: a critical review of the use of GPS telemetry data in ecology. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 365(1550), 2303-2312.

[2] Savalia, D. H. (2022). The Development of Smart Gas Distribution System *(Doctoral dissertation, School of Petroleum Management)*.

[3] Tamilarasi, K., & Jawahar, A. (2020). Medical data security for healthcare applications using hybrid lightweight encryption and swarm optimization algorithm. *Wireless Personal Communications*, 114, 1865-1886.

[4] Kanwal S. et al. (2021), Analytic Study of a Novel Color Image Encryption Method Based on the Chaos System and Color Codes, *Complexity*, doi.org/10.1155/2021/5499538

[5] Wang X. et al. (2022). A New V-Net Convolutional Neural Network Based on Four-Dimensional Hyperchaotic System for Medical Image Encryption", *Security and Communication Networks*, doi.org/10.1155/2022/4260804

[6] Bacara, C., Deville, D., Hauspie, M., & Grimaud, G. (2018, April). Challenges for the design of a privacy-preserving, multi-domain telemetry system for widely-spread network security appliances. *In Proceedings of the 1st Workshop on Privacy by Design in Distributed Systems* (pp. 1-6).

[7] Tariq, U., Ahmed, I., Bashir, A. K., & Shaukat, K. (2023). A Critical Cybersecurity Analysis and Future Research Directions for the Internet of Things: A Comprehensive Review. *Sensors*, 23(8), 4117.

[8] Doss, S., Paranthaman, J., Gopalakrishnan, S., Duraisamy, A., Pal, S., Duraisamy, B., & Le, D. N. (2021). Memetic optimization with cryptographic encryption for secure medical data transmission in IoT-based distributed systems. Computers, *Materials & Continua*, 66(2), 1577-1594.

[9] Liu, Y., Peng, H., & Wang, J. (2018). Verifiable Diversity Ranking Search Over Encrypted Outsourced Data. Computers, *Materials & Continua*, 55(1).

[10] Shah S.F.A. et al. (2024). Applications, Challenges, and Solutions of Unmanned Aerial Vehicles in Smart City Using Blockchain, *PeerJ Computer Science*, 10:e1776, doi.org/10.7717/peerj-cs.1776

[11] Haque, M. E., Zobaed, S. M., Islam, M. U., & Areef, F. M. (2018, December). Performance analysis of cryptographic algorithms for selecting better utilization on resource constraint devices. *In 2018 21st International Conference of Computer and Information Technology (ICCIT)* (pp. 1-6). IEEE.

[12] Seth, B., Dalal, S., Le, D. N., Jaglan, V., Dahiya, N., Agrawal, A. & Verma, K. D. (2021). Secure Cloud Data Storage System Using Hybrid Paillier–Blowfish Algorithm. *Computers, Materials & Continua*, 67(1).

[13] El-Shafai, W., Aly, M. H., Algarni, A. D., El-Samie, A., Fathi, E., & Soliman, N. F. (2022). Secure and Robust Optical Multi-Stage Medical Image Cryptosystem. *Computers, Materials & Continua*, 70(1).

[14] Abdul Hussien, F. T., & Aldeen Khairi, T. W. (2023). Performance Evaluation of AES, ECC and Logistic Chaotic Map Algorithms in Image Encryption. *International Journal of Interactive Mobile Technologies*, 17(10).

[15] Alemami, Y., Al-Ghonmein, A. M., Al-Moghrabi, K. G., & Mohamed, M. A. (2023). Cloud data security and various cryptographic algorithms. *International Journal of Electrical and Computer Engineering*, 13(2), 1867.

[16] Kanwal S. et al. (2022). An Effective Color Image Encryption Based on Henon Map, Tent Chaotic Map, and Orthogonal Matrices. *Sensors*. doi.org/10.3390/s22124359

[17] Sanap, S. D., & More, V. (2021, March). Analysis of encryption techniques for secure communication. *In 2021 International Conference on Emerging Smart Computing and Informatics (ESCI)* (pp. 290-294). IEEE.

[18] Kalaivani, V. (2023). Enhanced BB84 quantum cryptography protocol for secure communication in wireless body sensor networks for medical applications. *Personal and ubiquitous computing*, 27(3), 875.

[19] Ramachandra, M. N., Srinivasa Rao, M., Lai, W. C., Parameshachari, B. D., Ananda Babu, J., & Hemalatha, K. L. (2022). An efficient and secure big data storage in cloud environment by using triple data encryption standard. *Big Data and Cognitive Computing*, 6(4), 101.

[20] Ali, A., Al-Rimy, B. A. S., Alsubaei, F. S., Almazroi, A. A., & Almazroi, A. A. (2023). HealthLock: Blockchain-Based Privacy Preservation Using Homomorphic Encryption in Internet of Things Healthcare Applications. *Sensors*, 23(15), 6762.

[21] Yousif, S. F. (2023). Performance comparison between RSA and El-Gamal algorithms for Speech Data Encryption and decryption. *Diyala Journal of Engineering Sciences*, 123-137.

[22] Olutola, A., & Olumuyiwa, M. (2023). Comparative Analysis of Encryption Algorithms. *European Journal of Technology*, *7*(1), 1-9.

[23] Liu, J. (2023). Image security of agricultural environment monitoring based on image encryption algorithm of secure compressed sensing. *Journal of Biotech Research*, *14*, 185-195.

[24] Jabeen, T., Jabeen, I., Ashraf, H., Jhanjhi, N. Z., Yassine, A., & Hossain, M. S. (2023). An Intelligent Healthcare System Using IoT in Wireless Sensor Network. *Sensors*, *23*(11), 5055.

[25] Urooj, S., Lata, S., Ahmad, S., Mehfuz, S., & Kalathil, S. (2023). Cryptographic Data Security for Reliable Wireless Sensor Network. *Alexandria Engineering Journal*, *72*, 37-50.

[26] Aggarwal, G., & Sharma, M. H. (2023). Comparative Analysis of Multi-Level Algorithm with Different Encryption and Decryption Security Algorithms. *Tuijin Jishu/Journal of Propulsion Technology*, *44*(3), 1640-1648.

[27] Sabir, S., & Guleria, V. (2023). Multi-layer security based multiple image encryption technique. *Computers and Electrical Engineering*, *106*, 108609.

[28] Zheng, L., Wang, Z., & Tian, S. (2022). Comparative study on electrocardiogram encryption using elliptic curves cryptography and data encryption standard for applications in Internet of medical things. *Concurrency and Computation: Practice and Experience*, *34*(9), e5776.

[29] Raheja, N., & Manocha, A. K. (2022). IoT based ECG monitoring system with encryption and authentication in secure data transmission for clinical health care approach. *Biomedical Signal Processing and Control*, 74, 103481.

[30] Hameed, M. E., Ibrahim, M. M., Abd Manap, N., & Attiah, M. L. (2019). Comparative study of several operation modes of AES algorithm for encryption ECG biomedical signal. *International Journal of Electrical and Computer Engineering*, *9*(6), 4850.

[31] Dejene, D., Tiwari, B., & Tiwari, V. (2020). TD2SecIoT: temporal, data-driven and dynamic network layer based security architecture for industrial IoT. *IJIMAI*, *6*(4), 146-156.

[32] Tahir, M., Sardaraz, M., Mehmood, Z., & Muhammad, S. (2021). CryptoGA: a cryptosystem based on genetic algorithm for cloud data security. *Cluster Computing*, *24*, 739-752.

[33] Makarenko, I., Semushin, S., Suhai, S., Kazmi, S. A., Oracevic, A., & Hussain, R. (2020, October). A comparative analysis of cryptographic algorithms in the internet of things. In *2020 International Scientific and Technical Conference Modern Computer Network Technologies (MoNeTeC)* (pp. 1-8). IEEE.

[34] Stafford, Gary. (19July2020) "Environmental Sensor Telemetry Data." Www.kaggle.com,kaggle,www.kaggle.com/datasets/garystafford/environmental-sensor-data-132k. Accessed 11 Sept. 2023.

[35] Shafiq, M. et al. (2022). The Rise of "Internet of Things": Review and Open Research Issues Related to Detection and Prevention of IoT-Based Security Attacks. *Wireless Communications and Mobile Computing*. doi.org/10.1155/2022/8669348

[36] Nadeem, M., Arshad, A., Riaz, S., Wajiha Zahra, S., S Band, S., & Mosavi, A. (2023). Two layer symmetric cryptography algorithm for protecting data from attacks.

[37] Christina, L., & Joe Irudayaraj, V. S. (2014). Optimized Blowfish encryption technique. *International Journal of Innovative Research in Computer and Communication Engineering*, *2*(7), 5009-5015.

[38] Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., & Ferguson, N. (1998). Twofish: A 128-bit block cipher. *NIST AES Proposal*, *15*(1), 23-91.

[39] Sumartono, I., Siahaan, A. P. U., & Mayasari, N. (2016). An overview of the RC4 algorithm. *IOSR J. Comput. Eng*, *18*(6), 67-73.

[40] Paar, C., Pelzl, J., Paar, C., & Pelzl, J. (2010). The advanced encryption standard (AES). *Understanding Cryptography: A Textbook for Students and Practitioners*, 87-121.

[41] Mahdi, M. S., Hassan, N. F., & Abdul-Majeed, G. H. (2021). An improved chacha algorithm for securing data on IoT devices. *SN Applied Sciences*, *3*(4), 429.

[42] Singh, G., Kumar, A., & Sandha, K. S. (2011). A study of new trends in Blowfish algorithm. *International Journal of Engineering Research and Application*, *1*(2), 321-326.

[43] Alomari, M. A., Samsudin, K., & Ramli, A. R. (2009, May). A study on encryption algorithms and modes for disk encryption. *In 2009 International Conference on Signal Processing Systems* (pp. 793-797). IEEE.

[44] Fluhrer, S., Mantin, I., & Shamir, A. (2001). Weaknesses in the key scheduling algorithm of RC4. In *Selected Areas in Cryptography: 8th Annual International Workshop, SAC 2001 Toronto, Ontario, Canada, August 16–17, 2001 Revised Papers 8* (pp. 1-24). Springer Berlin Heidelberg.

[45] Ananya, B. L., Nikhitha, V., Arjun, S., & Gowda, N. C. (2023). Survey of applications, advantages, and comparisons of AES encryption algorithm with other standards. *International Journal of Computational Learning & Intelligence*, *2*(2), 87-98.

[46] Kebande, V. R. (2023). Extended-Chacha20 Stream Cipher With Enhanced Quarter Round Function. *IEEE Access*.

[47] Quirino, G. S., & Moreno, E. D. (2013). Architectural evaluation of asymmetric algorithms in ARM processors. *International Journal of Electronics and Electrical Engineering*, *1*, 39-43.

[48] Akram, M., Iqbal, M. W., Ali, S. A., Ashraf, M. U., Alsubhi, K., & Aljahdali, H. M. (2022). Triple Key Security Algorithm Against Single Key Attack on Multiple Rounds. *Computers, Materials & Continua*, *72*(3).

[49] Nakov, S., Stefanov, M., & Shideroff, M. Practical Cryptography for Developers (2018). *Dostupné z: https://cryptobook. nakov. com*.

[50] Wang, Z., Yao, Y., Tong, X., Luo, Q., & Chen, X. (2019). Dynamically reconfigurable encryption and decryption system design for the internet of things information security. *Sensors*, *19*(1), 143.

[51] Guerrero-Sanchez, A. E., Rivas-Araiza, E. A., Gonzalez-Cordoba, J. L., Toledano-Ayala, M., & Takacs, A. (2020). Blockchain mechanism and symmetric encryption in a wireless sensor network. *Sensors*, *20*(10), 2798.

[52] Chowdhary, C. L., Patel, P. V., Kathrotia, K. J., Attique, M., Perumal, K., & Ijaz, M. F. (2020). Analytical study of hybrid techniques for image encryption and decryption. *Sensors*, *20*(18), 5162.

[53] Shaheen, A. M., Sheltami, T. R., Al-Kharoubi, T. M., & Shakshuki, E. (2019). Digital image encryption techniques for wireless sensor networks using image transformation methods: DCT and DWT. *Journal of Ambient Intelligence and Humanized Computing*, *10*, 4733-4750.

[54] Yang, X., Xi, W., Chen, A., & Wang, C. (2021). An environmental monitoring data sharing scheme based on attribute encryption in cloud-fog computing. *Plos one*, *16*(9), e0258062.

[55] Li, M. (2022). Automatic Encryption Method of Sensor Network Capture Data Based on Symmetric Algorithm. *Wireless Personal Communications*, 1-15.

[56] Whitford, M., & Klimley, A. P. (2019). An overview of behavioral, physiological, and environmental sensors used in animal biotelemetry and biologging studies. *Animal Biotelemetry*, *7*(1), 1-24.