International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.1 (2025) pp. 899-910
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

Security of IoT Device and its Data Transmission on AWS Cloud by Using Hybrid Cryptosystem of ECC and AES

Neha KASHYAP[1]*, Sapna SINHA[2], Vineet KANSAL[3]

[1]Amity Institute of Information Technology, Noida, Uttar Pradesh, India
* **Corresponding Author Email:** kashyap9.neha@gmail.com - **ORCID:** 0000-0003-0368-1078

[2]Amity Institute of Information Technology, Noida, Uttar Pradesh, India
Email: ssinha4@amity.edu- **ORCID:** 0000-0002-2504-8030

[3]Institute of Engineering & Technology, AKTU, Lucknow, Uttar Pradesh, India
Email -vineet.kansal@yahoo.com - **ORCID:** 0000-0001-7918-2991

Article Info:

DOI: 10.22399/ijcesen.838
Received : 28 December 2024
Accepted : 06 February 2025

Keywords :

IoT,
AES,
ECC,
Raspberry Pi,
Cloud.

Abstract:

The expanding prevalence of the Internet of Things (IoT) and its devices presents significant security challenges mostly a lack of multi-factor authentication, light encryption, etc. This study uses Elliptical Curve Cryptography (ECC) and Advanced Encryption Standards (AES) to create a hybrid method with multiple security features for Raspberry Pi and data transmission on the cloud named Hybrid Cryptosystem ECC +AES. Data gathered and transferred to the cloud offers a faster and safer encryption mechanism on Raspberry Pi. This technique provides a notable gain in encryption performance over other previous algorithms by utilizing the speed of AES and the secure key exchange of ECC. The author developed a web application and implemented the algorithm for generating sample data, encryption, decryption processes, and uploading files to an Amazon Web Services (AWS) S3 bucket using Python programming which will benefit other IoT devices with limited memory and computational power.

1. Introduction

IoT devices have less memory area, which challenges implementing old encryption algorithms on it. To avoid these problems, we need a highly efficient algorithm to make our devices more secure. Since IoT devices work on the Internet, store sensitive data, and transfer this data to others, they are at a high risk of cyber-attack [1]. An efficient security algorithm can protect these devices and the data they transmit from unauthorized access. With the anticipated dramatic increase in the number of IoT devices in the upcoming years, the demand for secure communication and data protection will also rise [2]. So, efficient protected algorithms are required to meet this demand and ensure the security of IoT devices that can be useful with their growing numbers. IoT devices are used in different real-time applications such as smart cities, smart homes, transportation, healthcare, and manufacturing. Many secure algorithms will save these devices and the designed system from cyber-attacks [3,4]. IoT devices tend to collect and process huge amounts of data important for automation, decision-making, and analysis. Protecting this data is required to secure the privacy and sensitive information. Data security is one of the biggest challenges in the IoT environment, so a highly secure algorithm is required to protect the data from unauthorized access [5]. In summary, a highly secure and trusty algorithm is required for the security of IoT data and IoT devices that will be beneficial for cost-effectiveness and provide security from cyber threats. The cryptography method has various kinds of keys that help secure symmetric and asymmetric data. A symmetric key, also known as a private key, uses a single key to encrypt and decrypt data in cryptography [5]. Figure 1 represents the exchange of keys for encryption and the usage of symmetric and asymmetric cryptography between the sender and the receiver. Many algorithms like Advanced Encryption Standards (AES), Data Encryption Standards (DES), and Blowfish are used in symmetric cryptography that works in private keys. While the encryption performed by asymmetric keys needs two different keys for encryption and

decryption Elliptical Curve Cryptography (ECC) and Rivest- Shamir-Adleman (RSA) use the public key algorithms approaches. AES encryption algorithm is developed for securing sensitive data and government information across various areas. It is designed to work with 128-bit input blocks to improve the data security. ECC algorithms play an important role in different techniques that work on low-power smart devices of IoT, and it is good for supporting asymmetric cryptography [4-6]. The Raspberry Pi Foundation developed Raspberry Pi for the implementation of innovation in computer teaching, home robotics, and industrial automation. The size of the Raspberry Pi is small, like a debit card, but it can perform better than a regular computer [7]. The Raspberry Pi operating system is based on the Linux operating system and is useful for both 32-bit and 64-bit processors. Raspberry Pi can bring a web browser and a terminal [8-22]. In our research, we used Raspberry Pi 4 Model B.

Our research combines AES and ECC to provide a secure method for message encryption and decryption. ECC provides key pair generation and establishes a secure network communication channel while AES is used for symmetric encryption of the original message. This proposed system is developed using the Python Programming Language as a back end and HTML, and CSS to design the front-end web page that allows users to input plaintext messages, perform encryption and decryption, and upload files to an S# bucket of Amazon Web Services (AWS).

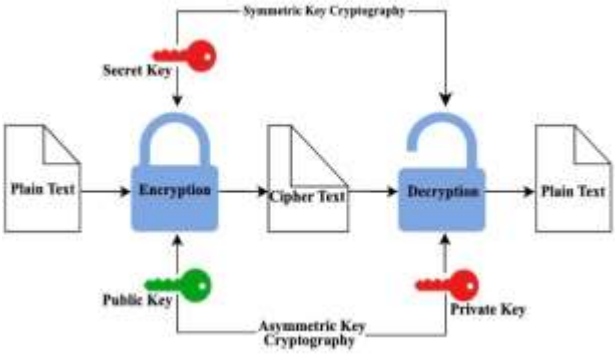


Figure 1. *Symmetric and Asymmetric Key Cryptography*

1.1. Motivation

The advanced use of IoT devices and their applications has become widespread in our everyday lives and data collection is sensitive and personal to users. For this scenario, it is essential to address security and privacy issues [1]. IoT network environment involves various components, a set of devices, various platforms, and implemented systems.

1.2. Contribution

The major contributions of this paper are as follows:

- Designing a cryptographic hybrid algorithm using a combination of ECC and AES for securing IoT devices and their data from unauthorized access and unwanted attacks.
- Generating the ECC private and public key and AES encrypted public key for the proposed hybrid ECC-AES cryptosystem scheme.
- Securing the Raspberry Pi and its collected data from unauthorized access with various levels of security.
- Examining the proposed method with various security aspects, computational efficiency, and time duration during data uploading on the cloud.

1.3. Outline of the Paper

This research paper structure is outlined as follows: Segment 2 details the literature review. Segment 3 represents the background descriptions of different aspects used in the proposed system. Segment 4 represents the details of key management of the proposed algorithm. Segment 5 represents the Architecture of the proposed work. Segment 6 presents proposed experimental setups. Segment 7 presents the proposed algorithm framework. Segment 8 presents the result analysis and discussion. Segment 9 shows the paper's conclusion

2. Literature Review

IoT devices' security and data security has become a major research area in today's era. Many algorithms, such as RSA, ECC, and AES are working for device security. M. Krishnamoorthy et.al (2017) [15] proposed that RSA is used for the initial setup and authentication of the network protocols within the slight changes. They suggested using ECC with RSA to create protocols that are resistant to the problem of factorization by employing quad prime numbers. The results of their work demonstrate that the proposed method offers high security with minimal computational costs. In a separate study, M. S. Albela (2018) [12] compared the Transport Layer Security authentication algorithm. They attempted to enhance IoT security by implementing wireless IoT nodes using ESP32. They utilized the ECC curve and RSA keys with varying lengths to secure the communication of IoT nodes. They attempted to enhance IoT security by implementing wireless IoT nodes using ESP32. They utilized the ECC curve and RSA keys with varying lengths to secure the communication of IoT nodes. H. Garg et al. (2019) [6] unequivocally demonstrate how connected devices can be securely exposed to cloud-based apps

and users with the Representational State Transfer (REST) API. In their proposed paradigm, middleware is critical in providing device data using REST, abstracting specifics, and acting as a user interface for interacting with sensor data. The proposed design incorporates non-IP networks to connect IoT devices, with an intelligent gateway connecting them to the Internet. Additionally, they recommend a straightforward JavaScript function to access a REST service at a given URL using jQuery's $. ajax () method, with the successful reception of JSON resulting in its assignment to the variable data. In 2020, Hassan et al. [9] worked on a cryptography algorithm, focusing on AES and ECC for encrypting multimedia contents and keys generated by ECC also shared between parties. They used Digital Rights Manager technology for the process of encrypting and storing the data in cloud environments. In 2021, Sowjanya et al. [20], proposed a process of key management for the CP-ABE mechanism using ECC. They worked on a revised method where the decryption process cannot be worked using the identical key unless text cannot be decrypted using the receiver having a different private key. Abdulhameed et al. (2022) [10] proposed a hybrid algorithm with the combination of AES and ECC to enhance the security of encryption and decryption process using Raspberry Pi Model 4. They worked with AES for data encryption and ECC for authentication. They also compare their proposed algorithm with previous works on the parameters of encryption time, decryption time, and throughput times. Urooj et.al. (2023) [11], worked on key management with the use of ECC and AES for security of data on the Wireless Sensor network (WSN).

3. Background

This segment describes the terms used in the background for the hybrid cryptosystem ECC+AES for security IoT data and devices.

3.1. Encryption

Encryption is the method used in cryptography to secure data and protect communication from unauthorized access [19]. The process of converting readable data, or plaintext, into unreadable data, or ciphertext, involves the use of a key and a specific encryption method [10]. The primary goal of encryption is to guarantee the confidentiality, integrity, and sometimes authentication of the data being transmitted or stored [4]. In our research, we used encryption for the security aspects of IoT data. The encryption method provides security for data stored on a Raspberry Pi and transferred to the cloud storage. Additionally, we encrypt the AES keys as an added layer of security.

3.2. Decryption

In cryptography, decryption is the method that is used to convert the encrypted data into its original and readable format. Encryption secures confidential and sensitive data by converting it into an incomprehensible format. Decryption helps restore the data to its original form and provide it to an authorized third party [7]. In this research, we used decryption of the ciphertext data and AES keys when uploaded to the cloud.

3.3. Advanced Encryption Standard (AES)

AES is a symmetric key method, The same key is used for both the encryption and decryption processes, its goal is to achieve a secure and effective design [10]. AES has three key sizes of 128, 192, and 256 bits are useful for providing security from brute-force attacks [16]. It is mostly used for the encryption of files, providing security to sensitive data and secure network communication [13]. In our research, we utilized AES for encryption on the Raspberry Pi because of its fast performance and resistance to hacking attempts due to its ability to work with longer key sizes. Figure 2 shows the AES design with a different key size.

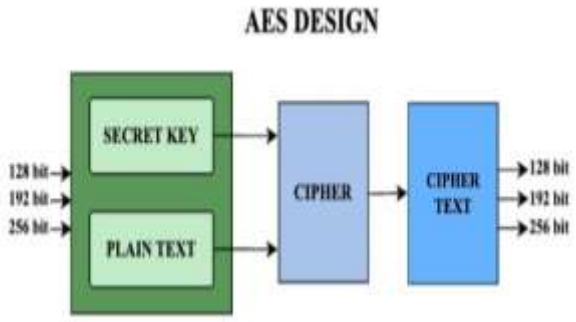


Figure 2. *AES Design with Different Key Sizes*

3.4. Elliptical Curve Cryptography (ECC)

One of the encryption techniques is ECC used in asymmetric public keys and utilizes the mathematical properties of elliptic curves on a limited field. It is getting used in secure web surfing, digital signatures, secure messaging, and secure communication offers high security despite its small key size. ECC is now the best selection for the network with less bandwidth [9]. In this research, we used ECC during the process of key exchange to encrypt the AES key to secure the data from unwanted attacks during data transmission to the cloud.

3.5. RealVNC Viewer Application

The company RealVNC offers remote access software. VNC stands for Virtual Network Computing. With the help of their VNC Connect software, the viewer can remotely manage the server's screen by exchanging data with the server via the Remote Frame Buffer (RFB) protocol. The viewer comprises two applications: VNC Server and VNC Viewer [23]. It was developed to provide remote control of another computer, and it is a cross-platform screen-sharing technology. This implies that a remote user can utilize a secondary device to operate a computer's keyboard, mouse, and screen from a secondary device as if they were seated in front of the computer [24]. Figure 3 shows the window screen of RealVNC. In this research, we use a VNC viewer for remote access to the device and work with generating data. Encryption, Decryption, and Uploading of data on the cloud.

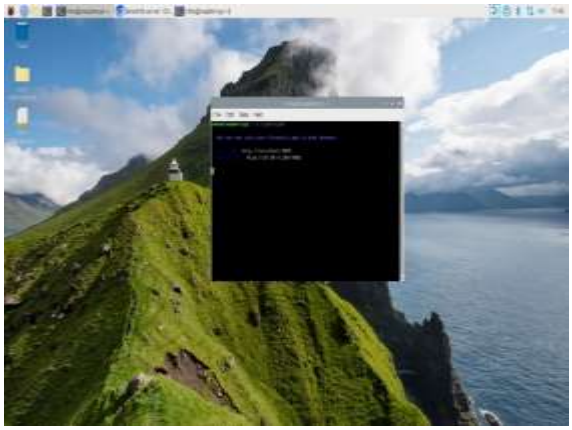


Figure 3. *RealVNC Viewer Application*

3.5. Raspberry Pi (RPi)

A small computer called Raspberry Pi is connected to a monitor, mouse, and Keyboard. For storing programs, it has different amounts of memory and a 1.5 GHz CPU for graphics processing [10]. Raspberry Pi uses ARM-based Embedded systems with the Linux operating system. It can be used for various tasks including IoT, programming, robotics, and media streaming [17]. This small computer allows users of all ages to learn and practice coding in different programming languages and experiment with programming [21]. It can do tasks like accessing the internet, creating spreadsheets or documents, and programming in popular languages like Python, C/C++, and Ruby [18]. Figure 4 shows the image of Raspberry Pi 4 Model B. Our research utilizes the Raspberry Pi 4 Model B, as depicted in Figure 3. Compared to the previous-generation RPi, the Raspberry Pi 4 Model B offers significant data storage, communication, multimedia capabilities, and CPU usage enhancements.

Figure 4. *Raspberry Pi 4 Model B*

3.6. Amazon Web Services (AWS) Simple Storage Service (S3) Bucket

AWS S3 is an object famous for storing data, providing security and, achieving high performance. It has features such as cost-effective storage and easy-to-use facilities. Its major uses are in the analysis of data, log files, application data, video, and pictures, and also in backup and recovery. Amazon S3 stores and protects big amounts of data for IoT devices, mobile applications, and big data analytics [25-30]. In our research, we used an AWS S3 bucket for the storage purpose of IoT data on the Amazon cloud. For this firstly we create the account on Amazon then create a bucket, given the bucket name, and AWS region. We also create an access key and download the access key ID and secret access key through the Identity and Access Management (IAM) console of AWS. After that, we upload our data to the bucket as an object in Amazon S3.

4. Key Management of Cryptosystem ECC+AES

In this segment, we are explaining the proposed schema management keys. Firstly, Using the AES method the data to transmitted in encryption. With the help of AES symmetric key encryption, which uses a shared key to both encrypt and decrypt the information. The encrypted data is then sent over a communication channel to the intended recipient. The recipient must have the secret key for encryption to decrypt the data. The ECC algorithm facilitates the secure exchange of this key between the sender and recipient. To generate the private and public keys, Elliptic curves are utilized in the ECC public-key encryption process. After receiving the public keys using randomly generated AES session keys, the sender can encrypt the message, and the other party receives the session key through the transmission route for encryption. The recipient

decrypts the session key using their private key. Once the recipient has the session key, they can decrypt the AES-encrypted material. After decrypting the data, the receiver can use it as necessary. Figure 5 shows the key management of the proposed scheme.

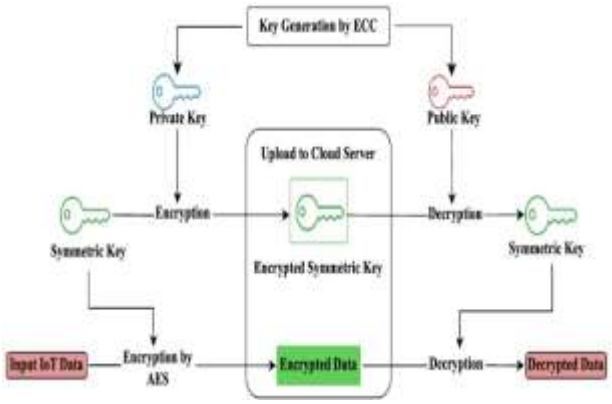


Figure 5. *Key Management of the Proposed Cryptosystem ECC+AES*

4.1. Role of ECC in Proposed Key Management

In the proposed system the ECC plays an important role in the encryption and decryption of IoT device-stored data. ECC also encrypts the AES key which has a long key length size. ECC is used in asymmetric cryptography, it works on an elliptical curve and provides a high level of security with its small key sizes. ECC generates two random key pairs: public key and private key. Figure 6 shows the front end of the proposed system that is written in Python Programming and deployed on Visual Studio Code for the key generation, encryption, and decryption of the data. The running scene of the proposed system on Visual Studio code is represented in Figure 7.

ECC Working Process for the Proposed System

Step 1: A second random ECC private key (‘ciphertextPrivKey’) is generated.

Step 2: This private key is used to compute the shared ECC key by multiplying it with the recipient’s public key (‘pubKey’). This shared key is an ECC point.

Step 4: The ECC point (x and y coordinates) is hashed using SHA-256 to derive the AES key.

Step 5: The ‘ecc_point_to_256_bit_key’ function takes this ECC point and derives a 256-bit key, which will be used as the AES secret key.

Step 6: The above-derived AES secret key is then used in AES-GCM mode to encrypt the message.

Step 7: The ‘encrypt_AES_GCM’ function encrypts the message(‘msg’) with an AES key and then it will return the data which is encrypted (ciphertext), a nonce, and an authentication tag(‘authTag’).

4.2. Role of AES in the Proposed Key Management

In the proposed system, the role of AES is to encrypt the data that are stored on the IoT devices and also to securely transfer it into the cloud environment. The AES key is not generated directly but is derived from the ECC shared key and used to encrypt the message with AES-GCM (Galois Counter Mode). The recipient can decrypt the message by deriving the same AES key from the shared ECC point.
AES Private Key Derivation and Encryption for the Proposed System

Step 1: The ECC private key (CiphertextPrivKey) is used to estimate the shared ECC key with the help of the receiver public key.

Step 2: The receiver's public key(pubKey) is used in conjunction with the receiver's private key to compute the shared ECC key.

Step 3: An elliptical point derived from the multiplication of the sender’s private key and the receiver’s public key. After that, it will be hashed to generate the AES key.

Step 4: The 256-bit key is then derived from the ECC shared point when used for AES-GCM (Galois/Counter Mode) encryption of the message.

5. Proposed Hybrid Cryptosystem ECC+AES Architecture

The suggested system includes the security framework for the IoT device application. ECC and AES are used for key generation and encryption of the stored data. This proposed system is written in Python and tested in Microsoft Visual Studio Code. The implemented system aims to secure the IoT device and store its encrypted data on the cloud. This goal is achieved by applying the algorithm that combines ECC and AES to provide a secure way of encrypting and decrypting messages and provide

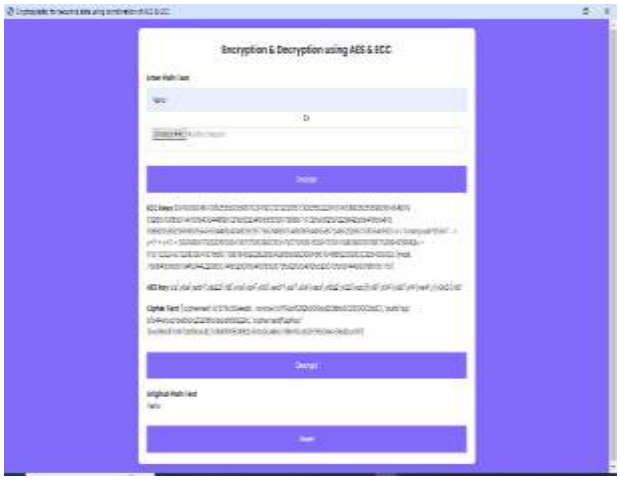


Figure 6. *Key Generation, Encryption, and Decryption of Proposed Cryptosystem ECC+AES*

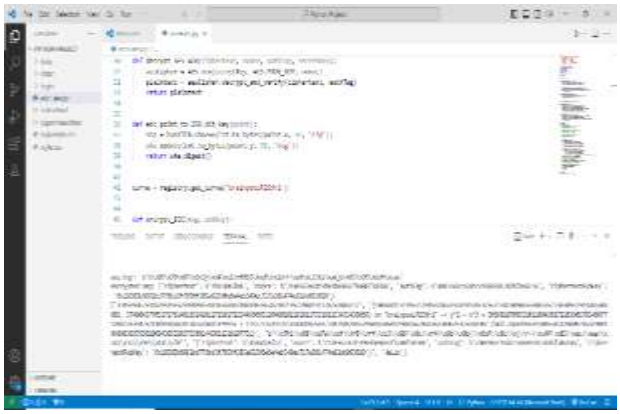


Figure 7. *Running Screen of Proposed Key Management of ECC and AES*

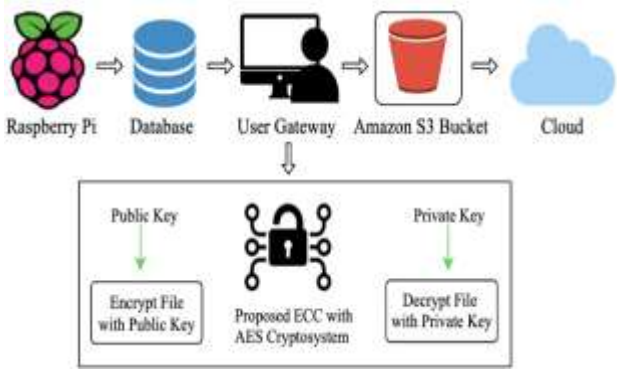


Figure 8. *Architecture of Proposed System*

security from brute-force attacks as AES has long keys. ECC generates key pairs of public and private. It sets up a secure communication channel, on the other hand, AES works on symmetric encryption of the original message. This web-based application allows users to input plaintext, messages, perform encryption and decryption, and save or upload files to an Amazon S3 bucket the entire working architecture of the proposed system is represented in Figure 8.

6. Experimental Setup

6.1. Set Up of Raspberry Pi

It is a low-cost computer plugged into a TV or monitor for display and uses a wired keyboard and mouse. We install the Raspberry Pi OS when we give power to Raspberry Pi, and all the peripherals boot into the freshly flashed Raspberry Pi OS (earlier known as Raspbian). There is no need for a CPU because Raspberry Pi itself is a microprocessor. To enhance the security of Raspberry Pi we use a two-factor authentication system that consists of device authentication and remote access authentication. Device authentication requires the security of Raspberry Pi with the combination of username and password. Remote access authentication is provided by RealVNC by creating a user account on the RealVNC website. The setup is shown in Figure 9.

Figure 9. *Setup of Raspberry Pi with TV, Keyboard, and Mouse*

Figure 10. *Set Up RealVNC with Raspberry Pi*

6.2. Set Up of RealVNC Viewer Software

RealVNC is a famous remote desktop software that provides access and control of Raspberry Pi with the help of other computers or mobile devices This software is used in our research for remote access of Raspberry Pi and shows the front end and back end of the system by uploading the entire proposed system project on Raspberry Pi. RealVNC has its authentication system for remote access, it allows the Raspberry Pi to securely connect over the

network. When we access Raspberry Pi remotely using the RealVNC Viewer, we have to give our RealVNC account credentials. Figure 10 is the set Up RealVNC with Raspberry Pi.

6.3. Set Up of the Proposed System of Hybrid Cryptography ECC+AES

The proposed system provides remote access to Raspberry Pi with the use of RealVNC. We installed the entire system written in Python programming and implemented the Visual Studio Code on the Raspberry Pi. When all the setups are ready and the authentication credentials are correct then the main design page will open with the options of "Generating Data", "Choose File", "Encrypt", and "Upload" as shown in Figure 11. Usage of the options used in the proposed system hybrid cryptosystem ECC+AES are as follows:

Generate Data
In the "Generate Data" tab, we created a sample CSV or text file with specified rows, columns, and tokens. It creates a random dataset. The maximum number of rows and columns we created through this tab is 1000 rows and 1000 columns, as shown in Figure 11(a).

Choose File
In the "Choose File" Tab, we uploaded a file in CSV or text format and entered plain text for encryption.

Encrypt
Through this tab, encryption is performed by using ECC and AES algorithms. The generated keys and encrypted message are displayed on the screen. Under this tab, the "Decrypt" button also shows the decryption of the ciphertext. Encryption and decryption time are also visible on the top of the page as shown in Figures 11(b) and 11 (c).

Upload
This tab uploads the encrypted file to the AWS S3 bucket cloud. We can view existing files and delete files from the bucket as shown in Figure 11(d).

7. Proposed Algorithm Of Hybrid Cryptosystem ECC+AES

The proposed algorithm shows the method of encryption and decryption of data for secure communication between sender and receiver. This proposed algorithm combines Asymmetric ECC and Symmetric AES to ensure the security of IoT data on Raspberry Pi and secure the data during transmission on the AWS cloud. Data is uploaded to the AWS cloud using the S3 bucket. Our algorithm has been compared to other algorithms and is more secure. Additionally, it is worth noting that Raspberry Pi is not commonly used for data generation. The algorithm of Data Encryption, Encryption of key, and Data Decryption methods are as follows:

7.2. Proposed Encryption Method

In this system, the encryption method is used for encrypting the generated data on Raspberry Pi and it also encrypts the AES key.

Algorithm 1: The Proposed Encryption Method
Input: plaintext: the data to be encrypted ecc_public_key: the public key for ECC encryption aes_key: the key for AES encryption **Output**: Encrypted Data **Step 1:** Generate random AES initialization vector (IV) **Step 2:** Encrypt plaintext using AES with generated IV and key **Step 3:** Encrypt AES key using ECC with public key **Step 4:** Concatenate encrypted AES key and ciphertext, separated by the IV **Step 5:** Output encrypted data

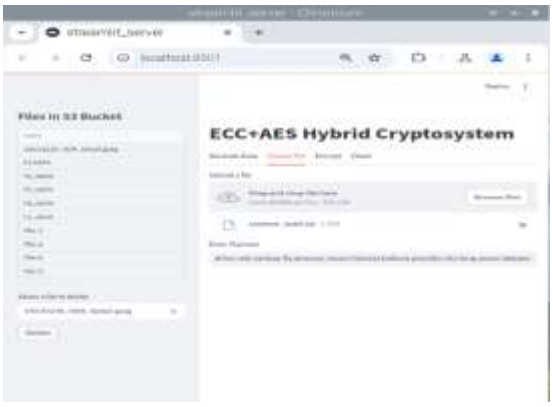
7.3. Proposed Decryption Algorithm

In this proposed system, the decryption algorithm is used to decrypt the ciphertext into its original text with the help of the AES symmetric key.

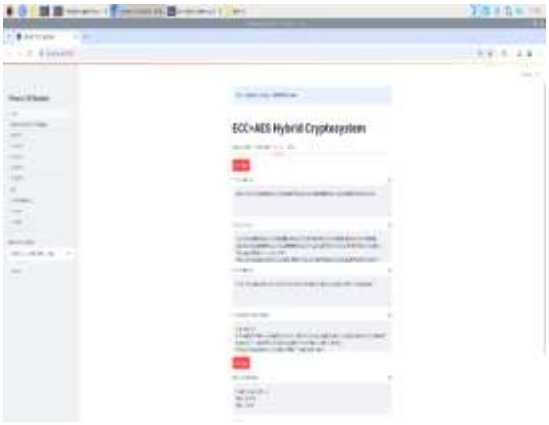
8. Performance Analysis and Discussion of Hybrid Cryptosystem ECC+AES

The proposed algorithm was implemented in the Python programming language and tested in the Visual Studio Code environment. The complete system was successfully deployed and operationalized on a Raspberry Pi showing its efficiency and practicality. The Suggested algorithm is compared with Hassan et.al. [9], who proposed two robust cryptosystems AES and ECC for the encryption and decryption of multimedia content that is implemented on cloud platforms. Their cryptographic method was implemented on a cloud platform and tested the adaptability and scalability of their proposed work. Abdulhameed et. al [10] proposed a hybrid algorithm for the encryption and decryption of data on Raspberry Pi and compared the time to encrypt and decrypt from previous related work of different authors.

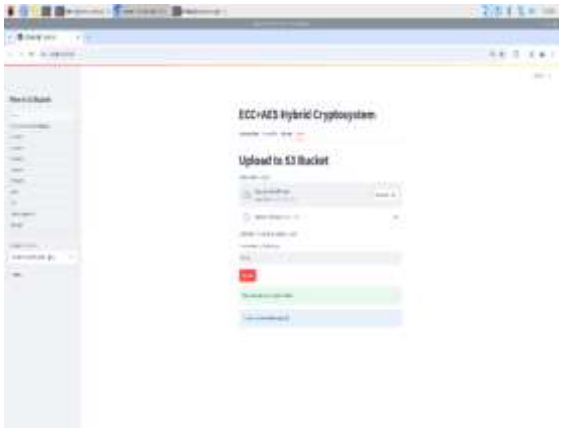
a). Generate Data Tab



b). Choose File Tab



c). Encrypt Tab



d). Cloud Tab

Figure 11. *Working on Various Tabs Used in the Proposed System*

Algorithm 2: Decryption Method

Inputs:
encrypted_data: the data to be decrypted
ecc_private_key: the private key for ECC decryption
Output: decrypted plaintext

Step 1: The first key_length bytes of encrypted_data are extracted as the encrypted AES key, and the next iv_length bytes are extracted as the initialization vector iv.

Step 2: The remaining bytes of encrypted_data are extracted as the ciphertext encrypted using AES.
Step 3: The AES key encryption encrypted_aes_key is decrypted using ECC decryption with the ecc_private_key. The resulting AES key is stored in aes_key.

Step 4: The encrypted ciphertext is decrypted using AES decryption with the aes_key and iv extracted in step 1. The resulting plaintext data is stored in plaintext.
Step 5: Finally, the plaintext is returned as the output of the decryption process.

Abdul Hammed et. al. used a Raspberry Pi device for encryption and decryption, but data was not uploaded to any cloud environment. Our innovative algorithm not only encrypts the public and private keys of Elliptic Curve Cryptography (ECC) but also secures the keys of Advanced Encryption Standard (AES). Our algorithm boasts faster encryption and decryption times compared to two other algorithms developed by different authors. Table 1 and Figure 5 provide a detailed comparison of the encryption times of our proposed algorithm with those of Hassan et al. [9] and Abdul Hammed et. al. [10]. Both author's groups did not use any AES key encryption to enhance the security of the device and data. Hassen et. al. does not use any IoT devices to implement their algorithms. Our proposed algorithm also encrypts the public and private keys of ECC as well as it encrypts the keys of AES. The proposed algorithm shows less encryption and decryption timing than the other two compared algorithms. Table 1 and Figure 12 shows the time taken by the proposed hybrid cryptosystem ECC+AES algorithm to encrypt data,

Table 1. *Comparison of Encryption, Decryption, and Uploading Time of Generated Data on RPi*

Size of File (CSV File in bytes)	No. of Rows & Columns	Encryption Time (in seconds)	Decryption Time (in seconds)	Uploading Time on the Cloud (in seconds)
6	1*1	0.169803	0.090174	0.153470
224	5*5	0.172257	0.132702	0.179805
793	10*10	0.191661	0.160461	0.183807
70100	100*100	0.274244	0.201486	0.344131
6900000	1000*1000	0.476506	0.362142	0.708058

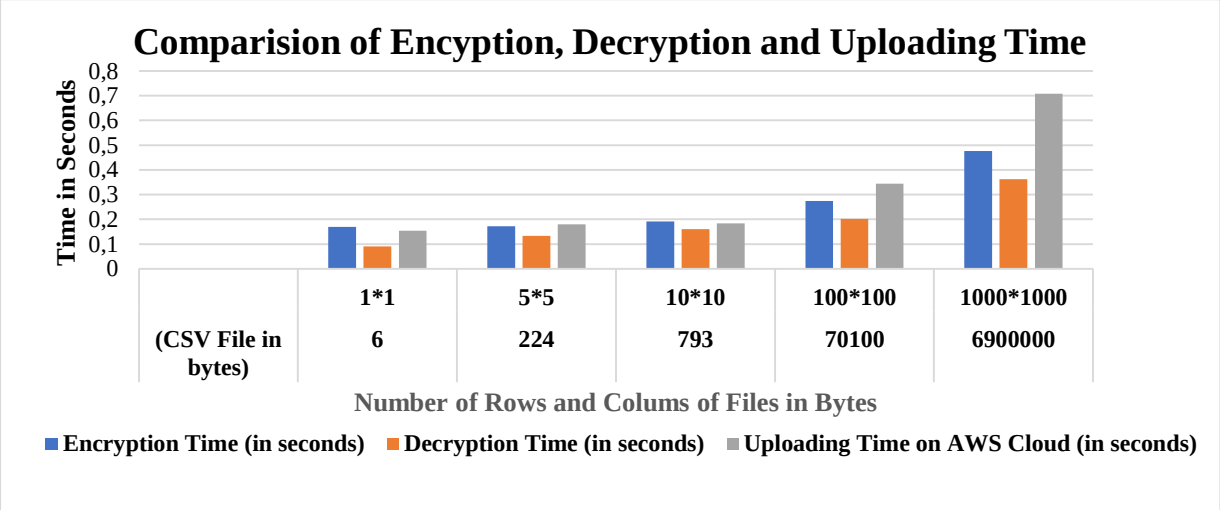


Figure 12. *Comparison of Encryption, Decryption, and Uploading Time*

Table 2. *Comparison of Encryption Time of Plaintext between the Previous and Proposed Presented Algorithms.*

Plaintext Size (Txt file in Bytes)	No. of Tokens	Hassan et.al [9]	Abdulhameed et. al [10]	Proposed Algorithm
17	3	0.120301	0.122636	0.114751
26	5	0.121542	0.129159	0.118202
64	10	0.138482	0.142619	0.133809
641	100	0.165732	0.168281	0.156681
6500	1000	0.174831	0.178308	0.171398

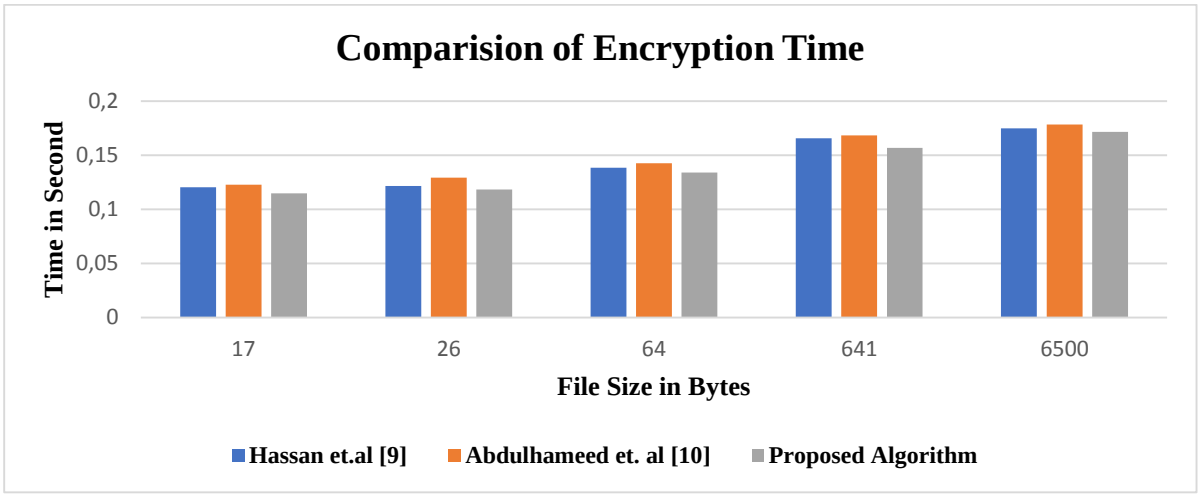


Figure 13. *Size of File vs Encryption Time*

Table 3. *Comparison of Decryption Time of plaintext between the Previous and Proposed Presented Algorithms.*

Plaintext Size (Txt file & bytes)	No. of Tokens	Hassan et.al [9]	Abdulhameed et. al [10]	Proposed Algorithm
17	3	0.118786	0.119649	0.105815
26	5	0.119659	0.121617	0.091441
64	10	0.134936	0.139392	0.096203
641	100	0.163571	0.163857	0.083261
6500	1000	0.171363	0.174753	0.083620

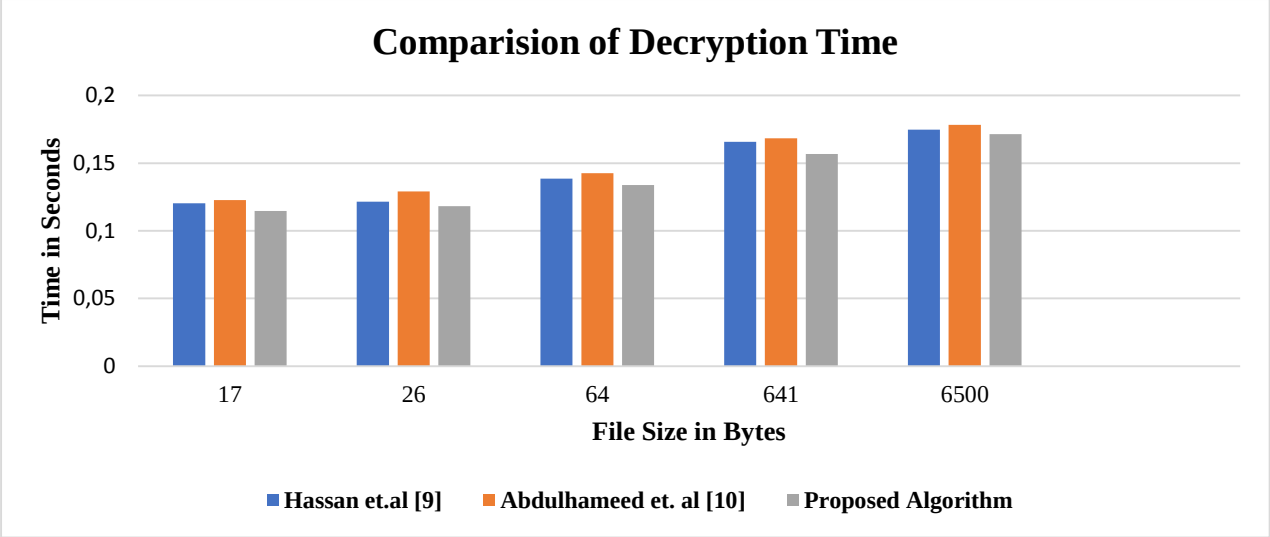


Figure 14. *Size of File vs Decryption Time*

decrypt data and upload data on Raspberry Pi. Table 2 and Figure 13 show the comparison of the encryption time of the proposed algorithm with Hassan et.al [9], and Abdulhameed et. al [10]. Table 3 and Figure 14 shows the comparison of the decryption time of the proposed algorithm with Hassan et.al [9], and Abdulhameed et. al [10].

9. Conclusions

In this research, we use the cryptographic algorithm of ECC+AES that helps to enhance the Security of IoT data on the Raspberry Pi and the cloud environment. Authors system encrypt the data or file and decrypt it. Files or data can be easily uploaded on the AWS S3 bucket. The Proposed algorithm is very secure on Raspberry Pi 4 Model B and easily encrypts, decrypts, and uploads files in very little time. After the result analysis and comparison with the previously designed AES-ECC-based algorithm it is found that our proposed algorithm is secure and user-friendly with various IoT devices as compared to others as shown in the comparison table and figure. In this system, the authorized user can only remotely access the Raspberry Pi after creating an account on Raspberry Pi and those who create their account on the S3 bucket of AWS can only upload the encrypted and decrypted data on this bucket. Finally, the main part of the system is that it secures IoT devices and generates data either in the form of text or in CSV. This system generates 1000 rows and 1000 columns of data and easily encrypts, decrypts, and uploads it.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data

are not publicly available due to privacy or ethical restrictions.

References

[1] N. Kashyap, A. Rana, V. Kansal and H. Walia (2021), "Improve Cloud Based IoT Architecture Layer Security - A Literature Review," *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, Greater Noida, India, pp. 772-777, doi: 10.1109/ICCCIS51004.2021.9397146.

[2] Kapito, B., Nyirenda, M., & Kim, H. (2021). Privacy-preserving Machine Authenticated Key Agreement for the Internet of Things. *International Journal of Computer Networks and Communications*, *13*(2), 99-120.

[3] Na, Y., Joo, Y., Lee, H., Zhao, X., Sajan, K. K., Ramachandran, G., & Krishnamachari, B. (2020, May). Enhancing the reliability of IoT data marketplaces through security validation of IoT devices. In *2020 16th International Conference on distributed computing in Sensor Systems (DCOSS)* (pp. 265-272). IEEE, DOI 10.1109/DCOSS49796.2020.00050

[4] Kashyap, N., Rana, A., Kansal, V., & Walia, H. (2021, September). Enhanced Data Security and Authentication Techniques for IoT Devices on Cloud. In *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)* (pp. 1-6). IEEE, doi: 10.1109/ICRITO51393.2021.9596232

[5] Kashyap, N., Sinhaa, S., & Kansal, V. (2024). A Hybrid Lightweight Method of ABE with SHA1 Algorithm for Securing the IoT Data on Cloud. *International Journal of Performability Engineering*, *20*(3). doi: 10.23940/ijpe.24.03.p1.131138

[6] Ashraf, Z., Sohail, A., & Yousaf, M. (2023). Robust and lightweight symmetric key exchange algorithm for next-generation IoE. *Internet of Things*, *22*, 100703, https://doi.org/10.1016/j.iot.2023.100703

[7] Rzepka, K., Szary, P., Cabaj, K., & Mazurczyk, W. (2024). Performance evaluation of Raspberry Pi 4 and STM32 Nucleo boards for security-related operations in IoT environments. *Computer Networks*, *242*, 110252, https://doi.org/10.1016/j.comnet.2024.110252

[8] Rajan, D. A. J., & Naganathan, E. R. (2022). Trust based anonymous intrusion detection for cloud assisted WSN-IOT. *Global Transitions Proceedings*, *3*(1), 104-108., https://doi.org/10.1016/j.gltp.2022.04.022

[9] Hassan, H. E. R., Tahoun, M., & ElTaweel, G. S. (2020). A robust computational DRM framework for protecting multimedia contents using AES and ECC. *Alexandria Engineering Journal*, *59*(3), 1275-1286., https://doi.org/10.1016/j.aej.2020.02.020

[10] Abdulhameed, H. A., Abdalmaaen, H. F., Mohammed, A. T., Mosleh, M. F., & Abdulhameed, A. A. (2022, March). A lightweight hybrid cryptographic algorithm for WSNs tested by the diehard tests and the raspberry Pi. In *2022 International Conference on Computer Science and Software Engineering (CSASE)* (pp. 271-276). IEEE., DOI: 10.1109/CSASE51777.2022.9759589

[11] 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., https://doi.org/10.1016/j.aej.2023.03.061

[12] Suárez-Albela, M., Fernández-Caramés, T. M., Fraga-Lamas, P., & Castedo, L. (2018, June). A practical performance comparison of ECC and RSA for resource-constrained IoT devices. In *2018 Global Internet of Things Summit (GIoTS)* (pp. 1-6). IEEE., DOI: 10.1109/GIOTS.2018.8534575

[13] Zodpe, H., & Sapkal, A. (2020). An efficient AES implementation using FPGA with enhanced security features. *Journal of King Saud University-Engineering Sciences*, *32*(2), 115-122, https://doi.org/10.1016/j.jksues.2018.07.002

[14] Challa, S., Das, A. K., Odelu, V., Kumar, N., Kumari, S., Khan, M. K., & Vasilakos, A. V. (2018). An efficient ECC-based provably secure three-factor user authentication and key agreement protocol for wireless healthcare sensor networks. *Computers & Electrical Engineering*, *69*, 534-554, http://dx.doi.org/10.1016/j.compeleceng.2017.08.003

[15] Krishnamoorthy, M., & Perumal, V. (2017). Secure and efficient hand-over authentication in WLAN using elliptic curve RSA. *Computers & Electrical Engineering*, 64, 552-566, http://dx.doi.org/10.1016/j.compeleceng.2017.06.002

[16] Williams, P., Dutta, I. K., Daoud, H., & Bayoumi, M. (2022). A survey on security in internet of things with a focus on the impact of emerging technologies. *Internet of Things*, *19*, 100564, https://doi.org/10.1016/j.iot.2022.100564

[17] Arreaga, N. X., Enriquez, G. M., Blanc, S., & Estrada, R. (2023). Security Vulnerability Analysis for IoT Devices Raspberry Pi using PENTEST. *Procedia Computer Science*, *224*, 223-230. https://creativecommons.org/licenses/by-nc-nd/4.0)

[18] Abas, S. U., Duran, F., & Tekerek, A. (2023). A Raspberry Pi based blockchain application on IoT security. *Expert Systems with Applications*, *229*, 120486. https://doi.org/10.1016/j.eswa.2023.120486

[19] Kumar, K., Ramkumar, K. R., & Kaur, A. (2022). A lightweight AES algorithm implementation for encrypting voice messages using field programmable gate arrays. *Journal of King Saud University-Computer and Information Sciences*, *34*(6), 3878-3885. https://doi.org/10.1016/j.jksuci.2020.08.005

[20] Sowjanya, K., Dasgupta, M., & Ray, S. (2021). A lightweight key management scheme for key-escrow-free ECC-based CP-ABE for IoT healthcare systems. *Journal of Systems Architecture*, *117*, 102108. https://doi.org/10.1016/j.sysarc.2021.102108

[21] Martínez-Fuentes, O., Díaz-Muñoz, J. D., Muñoz-Vázquez, A. J., Tlelo-Cuautle, E., Fernández-Anaya,

G., & Cruz-Vega, I. (2024). Family of controllers for predefined-time synchronization of Lorenz-type systems and the Raspberry Pi-based implementation. *Chaos, Solitons & Fractals*, *179*, 114462. https://doi.org/10.1016/j.chaos.2024.114462

[22] McBride, W. J., & Courter, J. R. (2019). Using Raspberry Pi microcomputers to remotely monitor birds and collect environmental data. *Ecological Informatics*, *54*, 101016. https://doi.org/10.1016/j.ecoinf.2019.101016

[23] Kisoon, R., Gumede, K., Fernandes, J., & Stopforth, R. (2024). Design of a remotely accessible satellite tracking system. In *MATEC Web of Conferences* (Vol. 406, p. 04012). EDP Sciences.

[24] Poorana Senthilkumar, S., Wilfred Blessing, N. R., Rajesh Kanna, R., & Karthik, S. (2024), Performance Evaluation of Predicting IoT Malicious Nodes Using Machine Learning Classification Algorithms. *International Journal of Computational and Experimental Science and Engineering*, *10*(3), 341-349. DOI: https://doi.org/10.22399/ijcesen.395

[25] Olariu, F., Alboaie, L., & Grămescu, R. (2024). Beyond Cloud Boundaries: An Analytical Case Study on the Migration of a Modular Monolith to Azure, AWS, and Google Cloud Platform. *Procedia Computer Science*, *246*, 2782-2791. https://doi.org/10.1016/j.procs.2024.09.393

[26] N. Vidhya, & C. Meenakshi. (2025). Blockchain-Enabled Secure Data Aggregation Routing (BSDAR) Protocol for IoT-Integrated Next-Generation Sensor Networks for Enhanced Security. *International Journal of Computational and Experimental Science and Engineering*, 11(1). https://doi.org/10.22399/ijcesen.722

[27] Amjan Shaik, Bhuvan Unhelkar, & Prasun Chakrabarti. (2025). Exploring Artificial Intelligence and Data Science-Based Security and its Scope in IoT Use Cases. *International Journal of Computational and Experimental Science and Engineering*, 11(1). https://doi.org/10.22399/ijcesen.869

[28] Alkhatib, A., Albdor , L., Fayyad, S., & Ali, H. (2024). Blockchain-Enhanced Multi-Factor Authentication for Securing IoT Children's Toys: Securing IoT Children's Toys. *International Journal of Computational and Experimental Science and Engineering*, 10(4). https://doi.org/10.22399/ijcesen.417

[29] P. Jagdish Kumar, & S. Neduncheliyan. (2024). A novel optimized deep learning based intrusion detection framework for an IoT networks. *International Journal of Computational and Experimental Science and Engineering*, 10(4). https://doi.org/10.22399/ijcesen.597

[30] Vutukuru, S. R., & Srinivasa Chakravarthi Lade. (2025). CoralMatrix: A Scalable and Robust Secure Framework for Enhancing IoT Cybersecurity. *International Journal of Computational and Experimental Science and Engineering*, 11(1). https://doi.org/10.22399/ijcesen.825