iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.78626

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

Cipherion - An Elliptic Curve Cryptography-Based System for File Transmission

Abhishek Mohanty[1], Areeb Khan[2], Nishant Dubey[3], Pawan Yadav[4], Prof. Raees Ahmed[5]
Computer Department, Theem College of Engineering, Boisar, India

Abstract: ***Cipherion is an Elliptic Curve Cryptography-based system designed for secure and efficient file transmission. While traditional methods like RSA provide security, they often impose high computational demands. Cipherion addresses this by utilizing Elliptic Curve Cryptography (ECC), which offers equivalent security to RSA with significantly lower overhead. The system employs a hybrid encryption approach: file contents are encrypted using symmetric algorithms (AES, DES, or ChaCha), while ECC facilitates the secure exchange of symmetric keys via the Elliptic Curve Diffie–Hellman (ECDH) protocol. Implemented in Python, the system features a modular design including file validation, key generation, and secure transfer components. Optimized for resource-limited environments such as IoT and mobile platforms, Cipherion provides a lightweight, scalable solution for robust file security.***

Keywords: AES, ChaCha, Cryptography, DES, ECDH, Elliptic Curve Cryptography (ECC), File Transfer System, Python.

I. INTRODUCTION

Cipherion – An Elliptic Curve Cryptography-Based System for File Transmission is a project developed to make file sharing more *secure and efficient*. In today's world, vast amounts of sensitive data are shared online, and protecting it during transfer is a major challenge. Traditional methods like RSA are strong, but they often impose *heavy computational demands*, especially for large files or on devices with limited resources. *Cipherion* addresses these challenges by utilizing *Elliptic Curve Cryptography (ECC)*, which provides the same level of security as RSA but with much *lower computational overhead*.

The system employs a *hybrid encryption approach* to protect files. The actual file content is encrypted with a fast symmetric algorithm such as *AES*, *DES*, or *ChaCha*, while ECC is used exclusively to protect the symmetric key. This architecture combines the *high-speed performance* of symmetric encryption with the *advanced security* of ECC, resulting in a lightweight and highly secure transmission process.

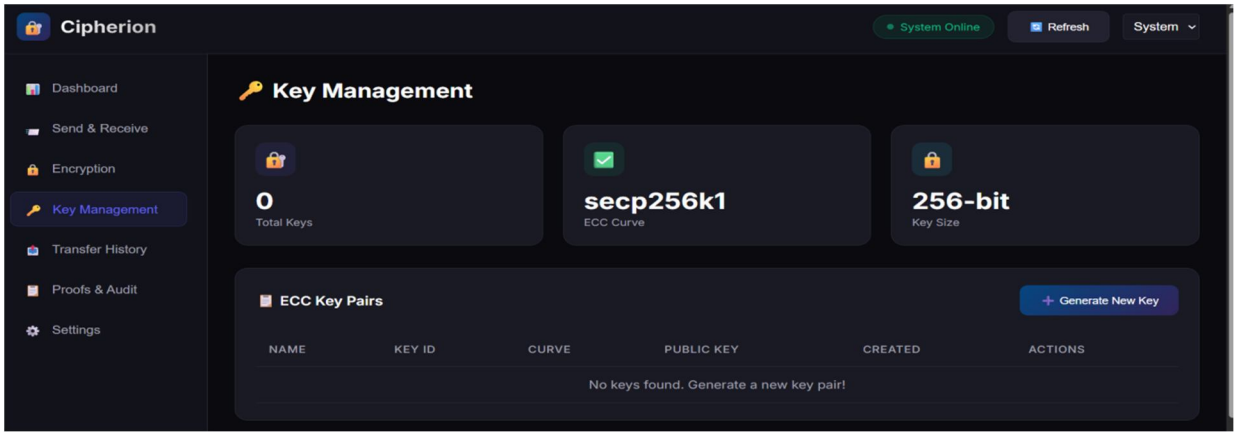


Fig. 1 Dashboard of Cipherion Secure File Transfer System

Fig. 1 above shows the main dashboard of the application. It has various fields showing the curve , the key size etc which are currently used for transferring the file. It also shows all the different menu options.

Implemented entirely in *Python*, the project leverages specialized cryptographic libraries such as *cryptography*, *eciespy*, and *ecdsa*. These tools facilitate the generation of ECC key pairs and the secure execution of the *Elliptic Curve Diffie–Hellman (ECDH)* protocol. The workflow is *modular and user-friendly*; files undergo strict *file validation* before encryption begins. This modular design ensures that *Cipherion* is easily *scalable* and maintainable for future cross-platform integration.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

II. LITERATURE REVIEW

The development of *Cipherion* is informed by a comprehensive analysis of *Elliptic Curve Cryptography (ECC)* and its application in securing digital communication. Current research emphasizes the transition from traditional RSA-based systems to *ECC* due to the latter's superior performance in *resource-constrained environments* [2, 10].

A. *Performance and Security of ECC*

Extensive studies demonstrate that *ECC* offers equivalent security to RSA but with significantly *smaller key sizes* [2, 8, 10]. For instance, a 256-bit *ECC* key provides the same security as a 3072-bit RSA key, leading to substantial savings in bandwidth, memory, and power consumption [6, 10]. Alhaj et al. further note that *ECC* operations are generally 2 to 4 times faster than RSA, making them ideal for mobile and IoT devices [2]. Research by Fang and Wu highlights the optimization of *scalar multiplication* using "double and add" methods to reduce time complexity from O(n) to O(log n) [5].

B. *Hybrid Cryptographic Implementations*

To balance high-level security with speed, several researchers propose *hybrid models*. Shaikh et al. introduced an algorithm combining symmetric ciphers like *AES* with *ECC* via the *Elliptic Curve Diffie-Hellman (ECDH)* protocol [4]. Similar architectures have been explored for *cloud computing* environments where *ECC* facilitates secure file sharing [11]. Zhou and Ding investigated *ECC* for secure exchange of chaotic initial keys in FPGA-based encryption chips [9].

C. *Encoding Schemes and Security Enhancements*

AlMajed and Almogren proposed the *SE-Enc* scheme, which utilizes a nine-phase process to ensure authenticated encryption, resisting *Known-Plaintext* and *Chosen-Ciphertext attacks* [1]. Kiran et al. proposed a framework integrating *ECC* with data distortion for privacy preservation in distributed data mining [3], while Sibal et al. explored hiding *ECC*-encrypted data within video frames using steganographic techniques [6].

D. *Research Gap and System Motivation*

Shah et al. identified complexities in parameter selection and coordinate systems—such as Affine versus Jacobian coordinates—that significantly impact computational costs in wireless sensor networks [10]. Vulnerabilities often arise from improper *message encoding* or a lack of unified *digital signatures* for non-repudiation [1, 7]. *Cipherion* addresses these gaps by implementing a modular *Python-based framework* that integrates *ECDH* for key exchange, *AES/ChaCha* for bulk encryption, and *ECDSA* for transaction authenticity.

III. PROPOSED SYSTEM ARCHITECTURE

The proposed system, *Cipherion*, is a comprehensive secure file transfer platform enabling confidential and authentic sharing of files [11]. *Cipherion* relies on *Elliptic Curve Cryptography (ECC)*, which provides robust security using comparatively smaller key sizes than RSA [2, 10], making it suitable for transmitting sensitive documents while optimizing speed and storage [8].

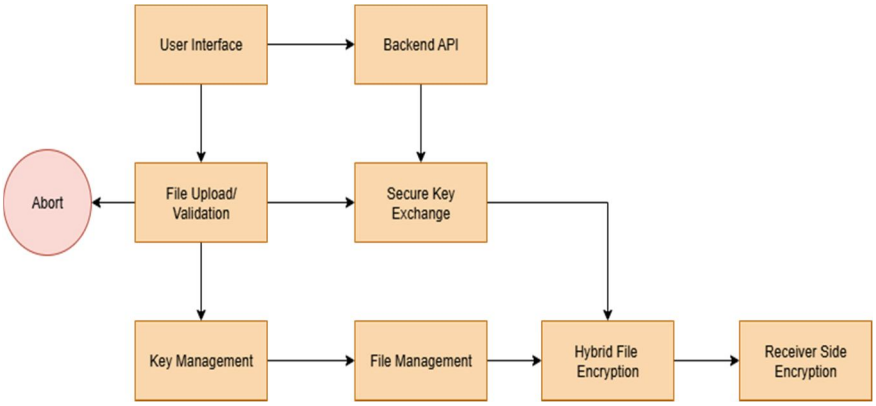


Fig. 2 System Architecture for Cipherion

Fig 2 above show out architecture for the proposed solution. We have all the components explained below.

A. System Components

The architecture of *Cipherion* adopts a hybrid cryptographic approach where *ECC* handles secure key establishment and symmetric encryption handles data confidentiality [4]. The framework consists of the following major modules:

1) User Interface and Backend API: The sender interacts via a secure interface supported by a Backend API using *Python* and Django to orchestrate business logic.
2) File Upload and Validation: This module verifies the integrity, type, and size of the file. Unauthorized requests trigger an Abort state to protect the system.
3) Key Management: Responsible for securely storing temporary keys and enforcing access controls for key retrieval.
4) Hybrid File Encryption Engine: Uses *AES* or *DES* for large file payloads; the session key is protected via *ECC* [4, 11].

B. Hybrid Encryption Workflow

The workflow follows a multi-stage process designed for end-to-end security:

1) Secure Key Exchange: Utilizing the ECDH protocol, parties combine the sender's public key and receiver's private key to derive a shared secret key [4, 9].
2) Message Signing: The system generates a digital signature using ECDSA, which binds the payload to the sender's identity [1, 8].
3) Receiver Decryption: The recipient verifies the digital signature using the sender's public key. Only upon successful verification is the shared secret used to decrypt the file [7].
4) Final Validation: The decrypted file undergoes a final integrity and malware scan before secure delivery.

IV. MATHEMATICAL FOUNDATIONS OF ECC

Elliptic Curve Cryptography (ECC) is a public-key encryption technique based on the algebraic structure of elliptic curves over finite fields. ECC provides a "trapdoor" function: computationally easy in one direction but virtually impossible to reverse, ensuring high security with minimal key sizes.

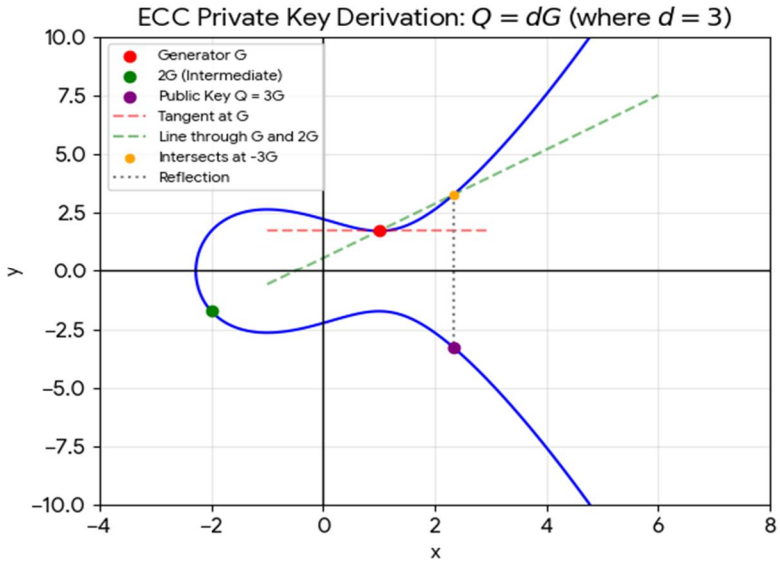


Fig. 3 ECC Graph – Short Weierstrass Curve

Fig 3 is the graph for the short weierstrass curve. It illustrates the geometric process of ECC private key derivation, specifically showing how a public key Q is calculated as the scalar multiple dG (where d = 3) on a Short Weierstrass curve. It visually demonstrates point addition and doubling, where a line is drawn through base points to find an intersection, which is then reflected across the x-axis to determine the resulting public key point.

A. The Elliptic Curve Equation

The security of *Cipherion* is built upon the Weierstrass equation, which defines the set of points (x, y) that constitute the curve. For a prime field GF(p), the equation is:

$$y^2 \equiv x^3 + ax + b \pmod{p} \quad \ldots (1)$$

To ensure the curve is non-singular (no cusps or self-intersections), the curve constants a and b must satisfy the discriminant condition:

$$4a^3 + 27b^2 \neq 0 \quad \ldots (2)$$

B. Point Arithmetic and Scalar Multiplication

The core cryptographic operation in *Cipherion* is scalar multiplication, which involves adding a generator point G to itself k times:

$$Q = k \cdot G \quad \ldots (3)$$

Where:

- *k* represents the randomly generated Private Key (a large integer).
- *G* is the Generator Point (a predefined base point on the curve).
- *Q* is the resulting Public Key (a point on the curve).

C. Elliptic Curve Diffie-Hellman (ECDH)

In *Cipherion*, ECDH derives a shared secret without transmitting it. If User A has private key d_a and public key Q_a, and User B has d^b and Q^b, the shared secret S is:

$$S = d_a \cdot Q^b = d_a \cdot (d^b \cdot G) = d^b \cdot (d_a \cdot G) = d^b \cdot Q_a \quad \ldots (4)$$

The resulting point S is passed through a Key Derivation Function (KDF) to generate the symmetric session key for bulk encryption [4].

D. Trapdoor Functionality

The security rests on the *Elliptic Curve Discrete Logarithm Problem (ECDLP)*. While computing Q from k and G is efficient, deriving k given only Q and G is computationally infeasible for large prime fields, forming the security foundation of all transmitted files.

V. EXPERIMENTAL SETUP

The performance and security of *Cipherion* were evaluated using a standardized hardware and software configuration to ensure reproducible results across various file formats and sizes [2, 10].

A. Hardware Specifications

Experiments were conducted on a system with an *Intel i5* or *AMD Ryzen 5* processor, *8 GB of RAM*, and a *512 GB SSD* to minimize I/O latency during bulk encryption. Network tests used a high-speed broadband connection to simulate real-world file transfer scenarios [10].

B. Software Environment

The Cipherion backend was developed in Python 3.8 with the Django framework. The cryptographic core used:

1) Cryptography and Pycryptodome: For symmetric algorithms (AES, ChaCha) and encryption primitives.
2) ECDSA: For generating and verifying digital signatures [1, 8].
3) Database Management: MySQL or *PostgreSQL* for persistent storage of user metadata and transaction logs.

C. Input Data and Benchmarking

Versatility was validated across plaintext (.txt), documents (.pdf), images (.jpeg), and multimedia files (.mp4), ranging from 81 kilobits to over 1 megabit [11]. Benchmarking used standard curves *Curve25519* and *SECP256k1* [8, 9].

TABLE I
PERFORMANCE ANALYSIS

File Type	Size (KB)	Enc. Time (ms)	Dec. Time (ms)
Text (.txt)	81	12.4	10.1
Image (.jpg)	450	45.8	41.2
Video (.mp4)	1024	112.5	108.3

Note: Performance data may vary depending on the deployment system.

D. Key Size and Security Equivalency

A primary advantage of *ECC* is its high security with shorter keys. As shown in Table II, the 256-bit key used in *Cipherion* offers security equivalent to a 3072-bit RSA key [2, 10] — a 91.6% reduction in key size that directly translates to lower memory consumption and faster transmission.

TABLE II
KEY SIZE SECURITY EQUIVALENCY

Security Level (Bits)	ECC Key (Bits)	RSA Key (Bits)
80	160–175	1024
112	224–255	2048
128	256–383	3072
256	512+	15360

E. Hybrid vs. Pure Asymmetric Encryption

While ECC is efficient for key management, encrypting large file payloads directly with ECC is computationally expensive. Table III shows processing time for a 1 MB file across different algorithmic combinations [4, 11].

TABLE III
ALGORITHMIC PERFORMANCE COMPARISON (1 MB FILE)

Algorithm	Type	Enc. Time	Security
RSA-2048	Asymmetric	450 ms	High
Pure ECC-256	Asymmetric	180 ms	Very High
ECC + AES-256 (Cipherion)	Hybrid	55 ms	Extreme
ECC + DES	Hybrid	75 ms	Moderate

F. Symmetric Algorithm Selection

AES is preferred over *DES* in the hybrid engine. *AES* uses a substitution-permutation network more resilient to differential cryptanalysis than *DES*'s Feistel network. Furthermore, *AES-256* provides a 128-bit security level, pairing perfectly with the 256-bit ECC key exchange to eliminate any "weak link" in the cryptographic chain [1].

G. Computational Efficiency

The hybrid ECC-AES model reduces CPU utilization by approximately 40% compared to pure RSA implementations, making it highly suitable for the resource-constrained environments discussed in Section II [8, 10].

VI. RESULTS AND DISCUSSION

The evaluation of *Cipherion* focuses on computational efficiency, throughput, and resilience against common cryptographic vulnerabilities. The results demonstrate that the hybrid approach provides a robust balance between security and performance [2, 4].

A. Performance Evaluation

The system was benchmarked on execution time and throughput across various file formats.

1) Encryption and Decryption Latency: Runtime for cryptographic operations remained minimal as file sizes increased. By offloading bulk encryption to symmetric ciphers and using *ECC* only for key encapsulation, the system avoids the exponential latency of pure asymmetric models [10].

2) Throughput Analysis: Throughput (measured in bps) remained stable for multimedia files up to 1 Mb, confirming that *Cipherion* handles real-time file sharing without significant bottlenecks [11].

B. Security Analysis

Security was validated against several attack vectors identified in the literature:

1) Resistance to Known Attacks: The SE-Enc encoding scheme and verified elliptic curves provide strong resistance against Known-Plaintext and Chosen-Ciphertext attacks [1, 8].

2) Integrity and Non-Repudiation: Integration of ECDSA ensures that any unauthorized modification of ciphertext during transit results in a verification failure at the receiver [4, 8].

3) Key Security: The ECDH protocol derives shared secrets locally without transmitting private keys, effectively mitigating man-in-the-middle (MITM) risks [4, 9].

C. Comparative Discussion

Consistent with Alhaj et al. and Shah et al., the implementation confirms that *ECC*-based systems outperform RSA in resource-constrained scenarios by providing equivalent security with 10% of the key size [2, 10]. The modular architecture allows independent optimization of field arithmetic and group operations, as suggested by Bai et al. [7].

D. System Screenshots

The following figures illustrate the Cipherion user interface and system components. [Insert figures from Images/UI/ and Images/Diagrams/ at the corresponding captions.

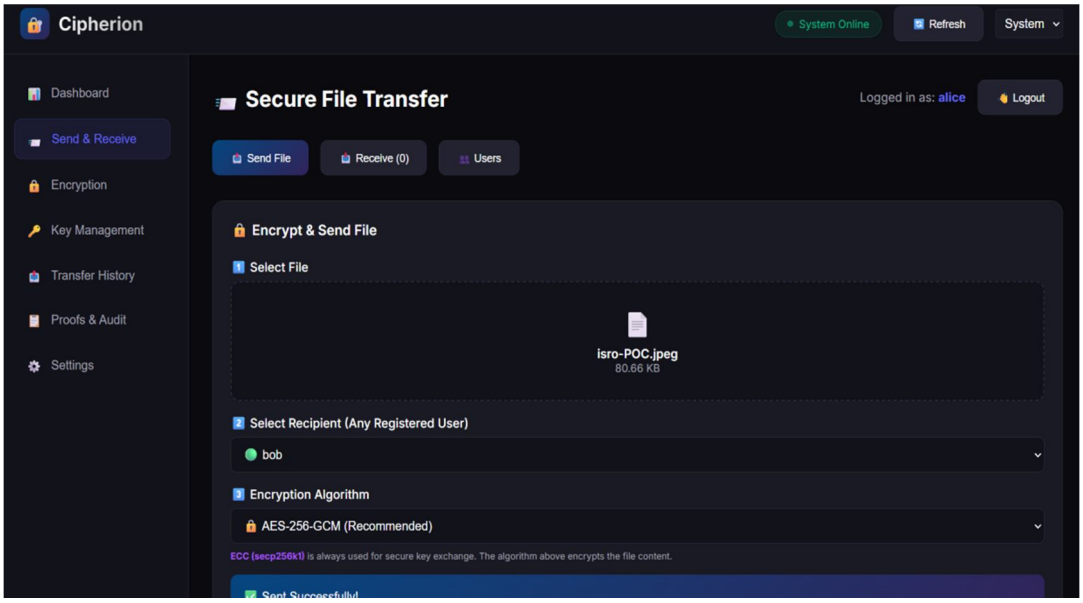


Fig. 4 Encryption Dashboard

The Fig 4 shows a successful file upload in the Cipherion dashboard, where a user named alice has selected an image file named isro-POC.jpeg for transfer. The interface confirms the recipient is bob and validates the use of AES-256-GCM for high-speed content encryption alongside ECC (secp256k1) for secure key management.

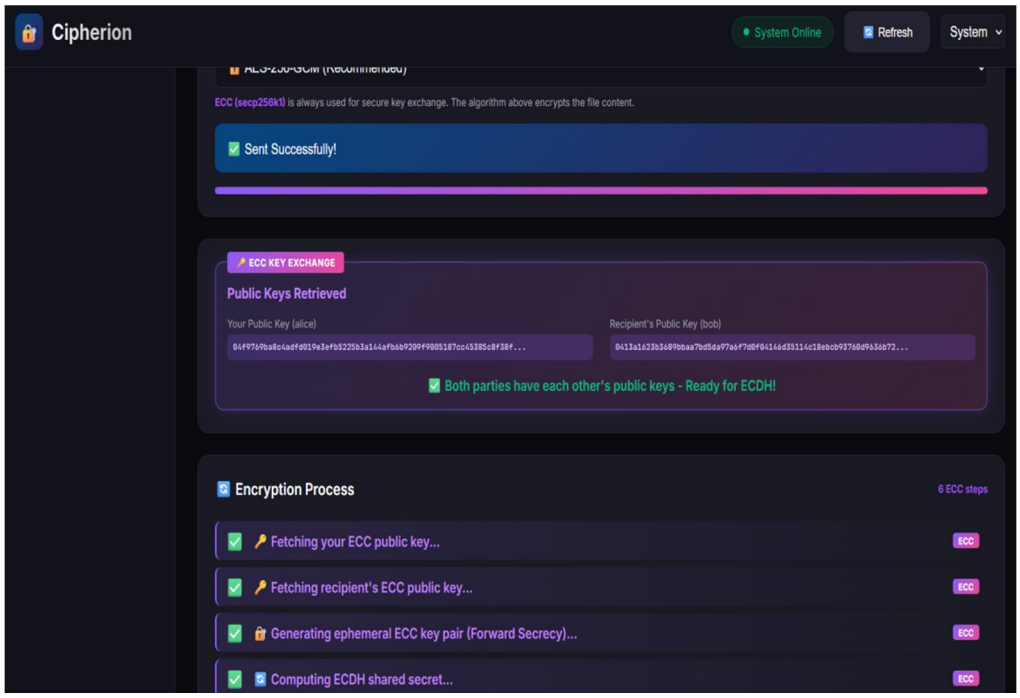


Fig. 5 Encryption Dashboard – Continue (Step 2)

The above fig 5 shows the ECC Key Exchange and Encryption Process within the Cipherion interface. It specifically highlights the retrieval of public keys between users (Alice and Bob) to prepare for the ECDH (Elliptic Curve Diffie-Hellman) handshake. The "Encryption Process" section tracks the progress of various security steps, such as fetching public keys, generating ephemeral keys for Forward Secrecy, and computing the shared secret.

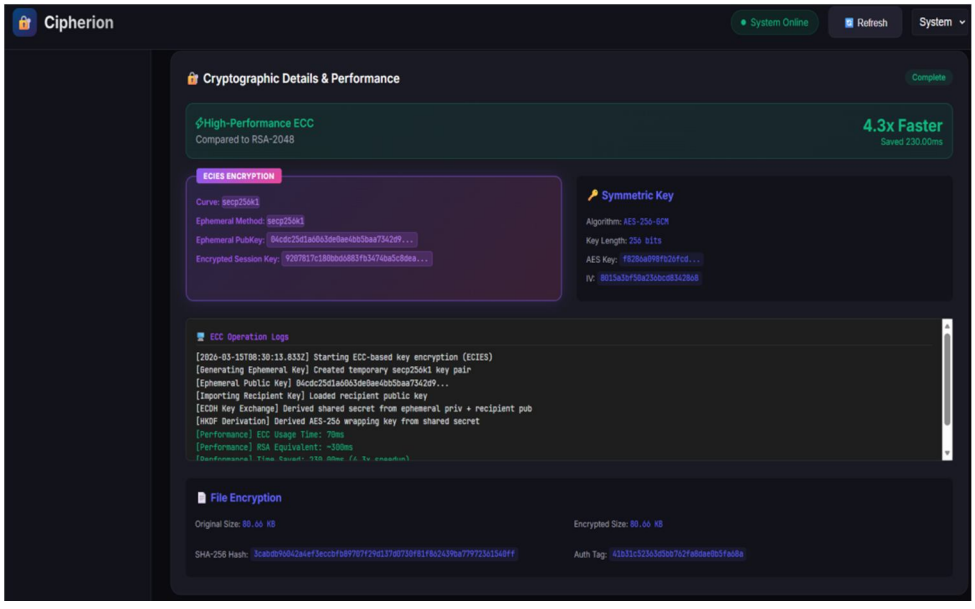


Fig. 6 Encryption Dashboard – Continue (Step 3)

This dashboard displays real-time cryptographic performance and technical logs for a file encryption task using the ECIES (Elliptic Curve Integrated Encryption Scheme) method. It highlights that the ECC-based process is 4.3x faster than RSA-2048, achieving a total time savings of 230.00 ms for the operation

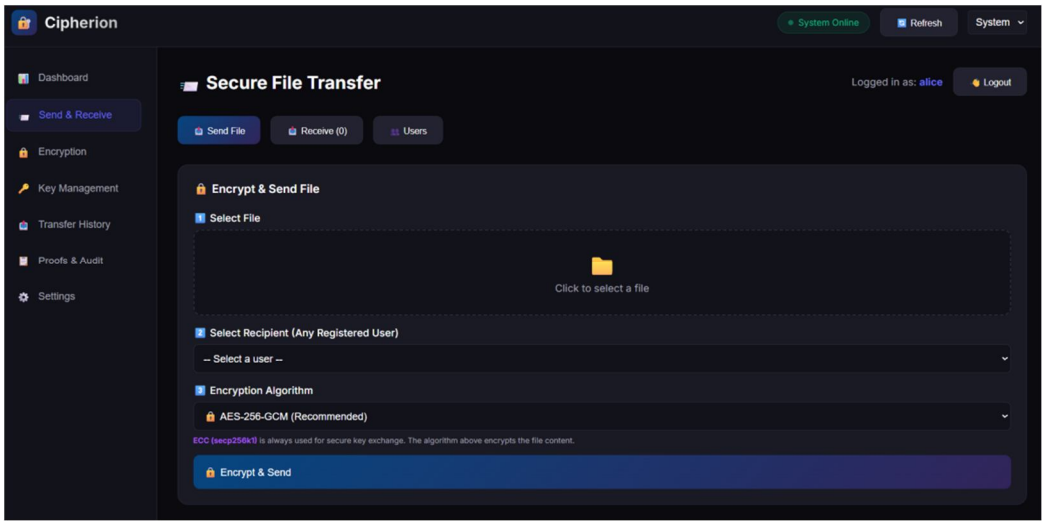


Fig. 7 Secure File Transfer Page

Fig 7 displays the Cipherion Secure File Transfer interface, which allows users to select a file and a registered recipient for encrypted transmission. The dashboard highlights the system's hybrid approach, specifically mentioning that ECC (secp256k1) is used for secure key exchange while offering symmetric algorithms like AES-256-GCM for the actual file content encryption.

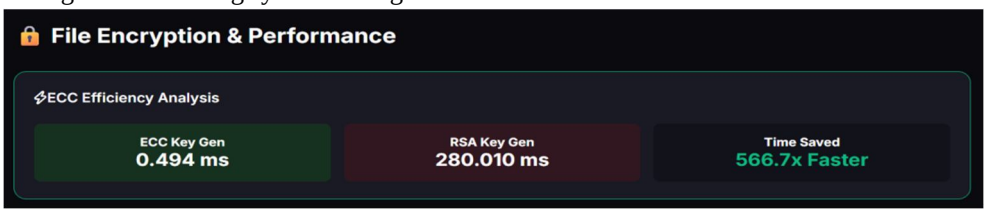


Fig. 8 Key Parameters Screen

The above fig 8 highlights an efficiency comparison where ECC key generation (0.494 ms) is 566.7x faster than RSA (280.010 ms). It demonstrates the significant performance advantage of Elliptic Curve Cryptography for rapid key establishment

VII. CONCLUSION AND FUTURE SCOPE

A. Conclusion

This research successfully designed and implemented *Cipherion*, a robust hybrid cryptographic framework for secure file sharing. By leveraging *ECC* for key management and symmetric ciphers for bulk encryption, the system achieves high-level security without the computational overhead of traditional RSA models [2, 10]. The integration of *ECDH* ensures secure local derivation of shared secrets, while *ECDSA* provides verifiable authenticity and non-repudiation [4, 8]. Experimental results confirm that the modular architecture reduces latency and maintains stable throughput across file formats, satisfying the requirements of confidentiality, integrity, and availability.

B. Future Scope

Several avenues for future research remain:

1) Hardware Acceleration: Offloading modular arithmetic to *FPGAs* or *ASICs* could further reduce power consumption in IoT environments [9].

2) Post-Quantum Readiness: Exploring lattice-based or other post-quantum primitives alongside ECC will be essential to maintain long-term security.

3) Cross-Platform Integration: Expanding into a distributed cloud ecosystem with multi-party computation would enable more scalable and decentralized secure data handling [11].

4) Steganographic Enhancements: Hiding encrypted payloads within high-definition video streams could provide an additional layer of covertness for sensitive data transfers [6].

VIII. ACKNOWLEDGMENT

The authors, Abhishek Mohanty, Areeb Khan, Nishant Dubey, and Pawan Yadav, express their sincere gratitude to the researchers whose pioneering work in Elliptic Curve Cryptography (ECC) provided the basis for this study. Spe cial thanks are extended to H. N. AlMajed and A. S. Almo gren for their insights into the SE-Enc encoding scheme [1], and to A. A. Alhaj, A. Alarbea, and M. Jawabreh for their analysis of efficient ECC transmission techniques [2].

The authors acknowledge the foundational work by P. Kiran, S. S. Kumar, and N. P. Kavya [3], as well as the hybrid security algorithms developed by A. P. Shaikh, V. Kaul, and S. K. Narayankhedkar [4]. We also recognize the contributions of X. Fang and Y. Wu [5], N. Sibal, T. Hasija, and S. Gupta [6], Q. H. Bai and colleagues [7], J. R. Shaikh et al. [8], and P. Zhou and Q. Ding [9]. Finally, we acknowledge the implementation studies of P. G. Shah, X. Huang, and D. Sharma [10], and the secure cloud research by V. Sharma, S. Chaudhary, and K. Singh [11].

REFERENCES

[1] H. N. AlMajed and A. S. Almogren, “Se-enc: A secure and efficient encoding scheme using elliptic curve cryptography,” IEEE Access, vol. 7, pp. 175865–175878, 2019.

[2] A. A. Alhaj, A. Alarbea, and M. Jawabreh, “Efficient and secure data transmission cryptography techniques using ecc,” Indonesian Journal of Electrical Engineering and Computer Science, vol. 36, pp. 486–492, Sep. 2024.

[3] P. Kiran, S. S. Kumar, and N. P. Kavya, “A novel framework using elliptic curve cryptography for extremely secure transmission in distributed privacy preserving data mining,” International Journal of Computer Applications, vol. 32, no. 10, pp. 85–95, 2012.

[4] A. P. Shaikh, V. Kaul, and S. K. Narayankhedkar, “Security enhancement algorithm for data transmission using elliptic curve diffie-hellman key exchange,” in Proc. Int. Conf. and Workshop on Advanced Computing (ICWAC), 2014.

[5] X. Fang and Y. Wu, “Investigation into the elliptic curve cryptography,” in Proc. 3rd Int. Conf. on Information Management (ICIM), pp. 412–415, 2017.

[6] H. Sibal, H. Hasija, and S. Gupta, “Secure transmission of data by elliptic curve cryptography,” Int. Journal of Engineering Research Technology (IJERT), vol. 5, no. 10, pp. 1–5, Oct. 2016.

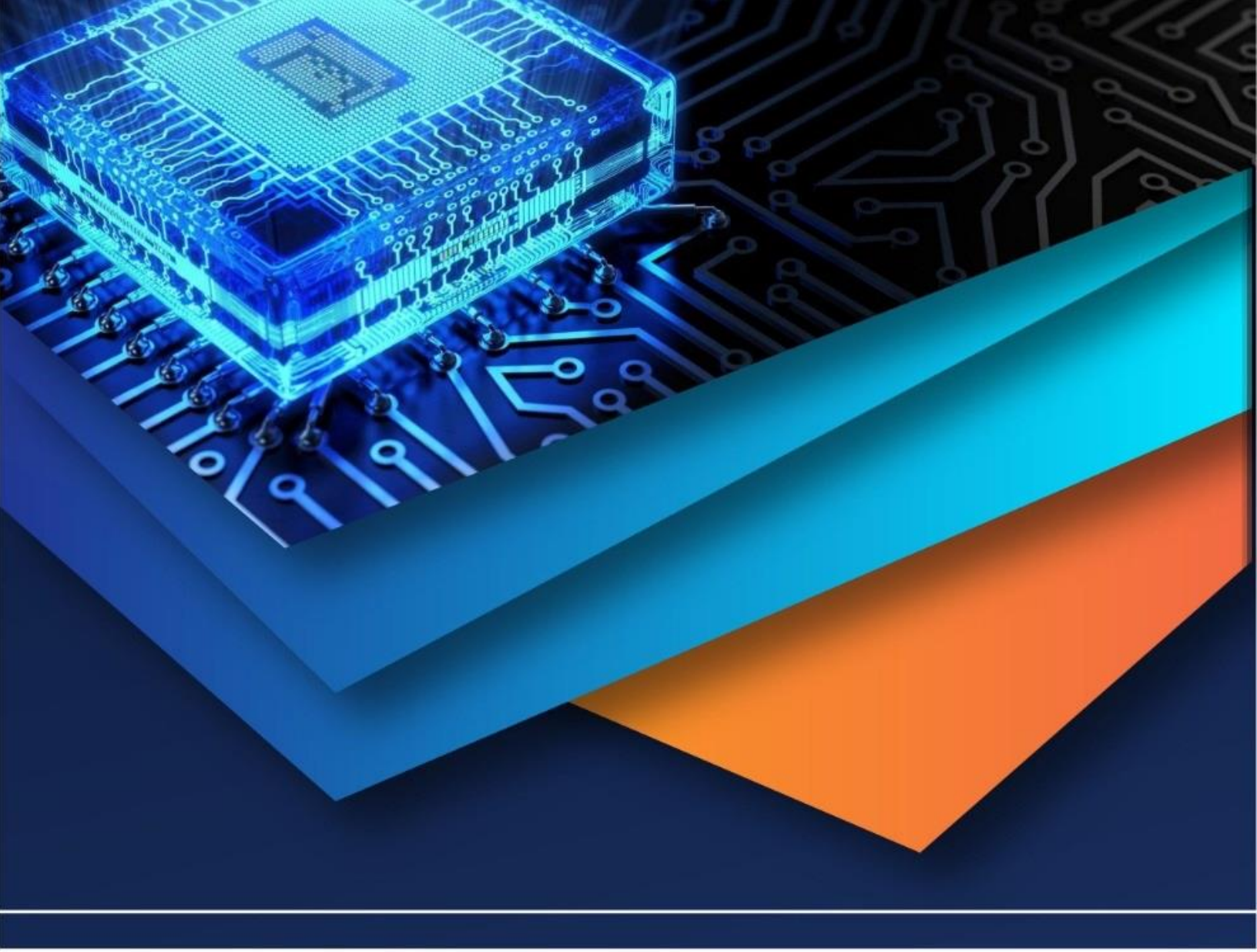
[7] Q. H. Bai, W. B. Zhang, P. Jiang, and X. Lu, “Research on design principles of elliptic curve public key cryptography and its implementation,” in Proc. 2nd Int. Conf. on Intelligent Computation Technology and Automation (ICICTA), pp. 880–883, 2009.

[8] J. R. Shaikh, M. Nenova, G. Iliev, and Z. Valkova-Jarvis, “Analysis of standard elliptic curves for the implementation of elliptic curve cryptography in resource-constrained e-commerce applications,” in Proc. IEEE Int. Conf. on Microwaves, Antennas, Communications and Electronic Systems (COMCAS), pp. 1–6, 2017.

[9] P. Zhou and Q. Ding, “Key exchange research of chaotic encryption chip based on elliptic curve cryptography algorithm,” in Proc. 2nd Int. Conf. on Intelligent Computation Technology and Automation (ICICTA), pp. 687–691, 2009.

[10] P. G. Shah, X. Huang, and D. Sharma, “Analytical study of implementation issues of elliptical curve cryptography for wireless sensor networks,” in Proc. IEEE 24th Int. Conf. on Advanced Information Networking and Applications Workshops (WAINA), pp. 589–592, 2010.

[11] V. Sharma, S. Chaudhary, and K. Singh, “Secure file sharing in cloud using elliptic curve cryptography,” Int. Journal of Advanced Computer Science and Applications (IJACSA), vol. 13, no. 3, pp. 458–463, 2022.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY