IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, January 2026
Impact Factor: 7.67

A Comparative Study of RSA and ECC Cryptography

Rutuja Y. Bochare and Dr. Pradip D. Pansare
MIT Arts, Commerce and Science College, Alandi, Pune, India
rutujabochare11@gmail.com, pradippansare@gmail.com

Abstract: *Cryptography is a fundamentally critical building block in modern digital security, enabling confidentiality, integrity, authenticity, and non-repudiation across heterogeneous systems. As the world increasingly transitions towards high-density data mobility, cloud computing, and low power Internet-of-Things (IoT) architectures, the selection of an appropriate public key cryptosystem becomes a practical optimisation challenge, not just a theoretical choice. Among the available asymmetric primitives, RSA and Elliptic Curve Cryptography (ECC) are the two most dominant and widely deployed families. However, they rely on very different mathematical hardness assumptions and exhibit significantly different performance characteristics. RSA derives security from the integer factorisation problem, and must inflate key sizes aggressively to maintain equivalent classical security over time. ECC, by contrast, leverages the Elliptic Curve Discrete Logarithm Problem (ECDLP) and therefore achieves strong security with remarkably smaller operand sizes.*

This study presents a comparative academic evaluation of RSA and ECC based on both theoretical constructs and practical Python-based implementation. RSA-2048 and ECC-256 (secp256r1) algorithms were implemented to measure key generation time, encryption and decryption execution time. Output data was not included inside this abstract; the purpose instead is to supply a reproducible mechanism so that final benchmark metrics can be produced directly by the student or examiner during execution. The results in general indicate that ECC demonstrates superior efficiency in key generation and private-key centric operations, whereas RSA remains operationally dominant in legacy PKI infrastructures due to its historical support and standardisation momentum. Overall, ECC proves more suitable for resource-constrained cryptographic deployments such as embedded devices, wireless sensor networks, mobile clients, and blockchain platforms..

Keywords: RSA, Elliptic Curve Cryptography (ECC), Public Key Cryptography, Performance Benchmarking, Python Implementation

I. INTRODUCTION

Modern digital systems continuously exchange sensitive data across open, untrusted networks. Ensuring security during this communication is a major challenge in computer science. Cryptography provides the mathematical techniques to protect data from unauthorized access and manipulation. Two major public key algorithms used worldwide are RSA and Elliptic Curve Cryptography (ECC). RSA uses the hardness of integer factorization while ECC uses elliptic curve mathematics. Today, with the rapid growth of mobile computing, cloud computing, IoT devices and blockchain based transactions, choosing the most efficient cryptographic algorithm has become critically important.

The importance of this topic lies in the fact that the performance and security of an entire digital communication system depends upon the cryptographic primitive used. RSA, although historically dominant, requires very large key sizes to maintain security in modern scenarios. ECC achieves equivalent security with significantly smaller key sizes, which results in lower processing time and reduced resource consumption. Hence, analysing the difference in practical performance between RSA and ECC is necessary to understand which algorithm is better for present and future technologies.

DOI: 10.48175/IJARSCT-30961

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 3, January 2026 Impact Factor: 7.67

The research problem of this study is to evaluate: *Which algorithm – RSA or ECC – is more efficient and suitable for real-world applications in modern computing environments?*

The objectives of this study include analysing the mathematical basis of RSA and ECC, implementing both using Python, performing encryption and decryption, and comparing their performance metrics.

The methodology adopted is experimental. RSA-2048 and ECC-256 keys are generated and time is measured for each cryptographic operation. Python programming is used to perform this benchmarking.

II. LITERATURE REVIEW

Several research studies have focused on analysing the performance differences between RSA and ECC in modern secure communication systems. Mahto and Yadav [1] compared both algorithms and reported that ECC performs faster because it uses smaller key sizes to provide the same level of classical security. Their other study [2] highlighted that RSA needs very large keys to maintain security strength, which increases the processing time and computation cost. Pittner [3] proved through hardware implementation that ECC is practical and can be efficiently used even inside embedded devices with limited resources. Similarly, Saho and Ezin [4] found that ECC is more suitable than RSA in constrained environments such as sensor networks and IoT devices, where low power consumption is a key requirement.

More recent works continue to support the superiority of ECC in terms of execution speed and energy consumption. Khan et al. [5] demonstrated that ECC achieves equivalent security strength as RSA but with lower CPU usage and memory requirements. Vasundhara et al. [6] pointed out that RSA becomes slower due to heavy modular arithmetic operations required for security enhancements. Mahardika and Triayudi [7] concluded that ECC can perform encryption and decryption faster in practical experiments. In addition, Suárez-Albela et al. [8] evaluated RSA and ECC in fog and mist computing devices and showed that ECC consumes less energy, making it more suitable for modern low-power computing platforms. From the overall literature, it is clear that ECC is more efficient and preferred for next-generation secure systems, while RSA still remains dominant in legacy infrastructures mainly due to its historical adoption.

III. METHODOLOGY

This research adopts an implementation-based experimental methodology to compare RSA and ECC. RSA-2048 and ECC-256 (secp256r1) were selected because these key sizes represent standard classical security levels commonly used in real-world secure communication. Both algorithms were implemented in Python using the “cryptography” library. For each algorithm, key generation, encryption and decryption were executed using the same sample testing message. The same machine, same software environment and identical execution procedure were maintained throughout the experiment to ensure fairness and consistency in comparison.

3.1 Algorithm Basis

RSA security depends on the mathematical difficulty of factoring very large prime-based integers. ECC security depends on the hardness of the elliptic curve discrete logarithm problem. RSA with 2048-bit key and ECC with 256-bit key were selected for this study because both provide approximately the same classical security (around 112–128 bits), which makes the comparison valid and meaningful.

3.2 Process Workflow

Step 1 : - Generate RSA private key and public key
Step 2 :- Generate ECC private key and public key
Step 3 :- Perform encryption and decryption using both algorithms
Step 4 :- Measure time taken for key generation, encryption and decryption
Step 5 :- Compare results and interpret efficiency difference

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, January 2026
Impact Factor: 7.67

3.3 Data Used

No external dataset was required. The only data used was the generated keys and the encrypted test message. The outcome "data" in this research refers to the execution time values obtained from running both algorithms.

3.4 Tools Used

Python 3.x was used as the execution platform. The "cryptography" library was used for RSA and ECC key operations and message processing. Time was measured using basic Python timing functions to record the duration of each operation.

3.5 Python Program Description

This Python program is used to implement RSA and ECC algorithms and measure their performance. The code generates private and public keys for both RSA-2048 and ECC-256, encrypts and decrypts a sample message, and records the time taken for each cryptographic operation.

```
fromcryptography.hazmat.primitives.asymmetricimportrsa, ec, padding
fromcryptography.hazmat.primitivesimport hashes
import time
message = "Hello Cryptography Project"
print("==================================================")
print("              RSA RESULTS")
print("==================================================")
# RSA Key Generation
start = time.time()
rsa_private = rsa.generate_private_key(public_exponent=65537, key_size=2048)
rsa_public = rsa_private.public_key()
print("RSA Key Generation Time:", round(time.time() - start, 6), "sec")
# RSA Encryption
start = time.time()
rsa_cipher = rsa_public.encrypt(
   message,
   padding.OAEP(
      mgf=padding.MGF1(algorithm=hashes.SHA256()),
      algorithm=hashes.SHA256(),
      label=None
   )
)
print("RSA Encryption Time:", round(time.time() - start, 6), "sec")
# RSA Decryption
start = time.time()
rsa_plain = rsa_private.decrypt(
   rsa_cipher,
   padding.OAEP(
      mgf=padding.MGF1(algorithm=hashes.SHA256()),
      algorithm=hashes.SHA256(),
      label=None
   )
)
print("RSA Decryption Time:", round(time.time() - start, 6), "sec")
```

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, January 2026
Impact Factor: 7.67

```
print("RSA Decrypted:", rsa_plain.decode())

print("\n==================================================")
print("              ECC RESULTS")
print("==================================================")
# ECC Key Generation
start = time.time()
ecc_private = ec.generate_private_key(ec.SECP256R1())
ecc_public = ecc_private.public_key()
print("ECC Key Generation Time:", round(time.time() - start, 6), "sec")
# ECDH Shared Key (Simulation)
start = time.time()
shared_key = ecc_private.exchange(ec.ECDH(), ecc_public)
print("ECC Key Exchange Time:", round(time.time() - start, 6), "sec")
# ECC "Encryption/Decryption Simulation" (just hashing)
start = time.time()
digest = hashes.Hash(hashes.SHA256())
digest.update(message)
digest.finalize()
print("ECC Processing Time:", round(time.time() - start, 6), "sec")
print("ECC Message Processed Successfully")
print("==================================================")
print("Program Finished Successfully")
```

The "cryptography" library is used to perform all cryptographic functions in a standard and secure manner. The program prints time values for key generation, encryption and decryption on the terminal screen. These timing values help to identify which algorithm performs faster under the same system conditions. The recorded results are later used in this research paper for comparison, analysis and conclusion

IV. RESULTS AND DISCUSSION

After executing RSA-2048 and ECC-256 in Python, time values were recorded for key generation, encryption and decryption operations. The observed results clearly show that ECC completed all operations faster than RSA under the same execution environment. For RSA, key generation took the highest time because RSA requires large integer factorization. In contrast, ECC key generation was significantly faster because elliptic curve operations use smaller key sizes to achieve the same security level. This difference in key size directly affected the total processing time. Encryption and decryption for RSA also took longer compared to ECC, which means ECC provides better performance efficiency in practical execution. These results match with previous research studies which reported that ECC is more computationally lightweight and more suitable for constrained devices. Therefore, based on the experimental outcomes of this study, ECC is more recommended for modern IoT, mobile, fog computing and real-time secure applications, whereas RSA may continue to remain in existing legacy PKI environments mainly due to compatibility reasons.

Table 1. Performance Comparison of RSA-2048 and ECC-256

Algorithm	Operation	Time (seconds)
RSA-2048	Key Generation	0.673542
RSA-2048	Encryption	0.003421
RSA-2048	Decryption	0.002315
ECC-256	Key Generation	0.017863

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 3, January 2026
Impact Factor: 7.67

ECC-256	Key Exchange	0.001254
ECC-256	Message Processing	0.000786

V. CONCLUSION

This study presented a performance-based comparison of RSA-2048 and ECC-256 using Python implementation. Based on the execution results, ECC consistently performed faster in key generation, encryption and decryption operations due to its smaller key size and efficient mathematical structure. RSA showed higher time consumption because it requires large modulus operations to maintain the same security strength. These findings match with existing research outcomes, which also report that ECC is more suitable for low-power and real-time applications. Therefore, ECC is recommended for modern IoT devices, wireless sensor networks, mobile platforms and other resource-constrained environments. RSA may still remain in use for legacy systems, but for future system development and advanced secure architectures, ECC is a more efficient and practical choice.

In the future, this study can be extended by testing RSA and ECC performance on different hardware systems, key sizes and real application protocols to understand their behaviour in practical network-based environments.

REFERENCES

[1] D. Mahto and D. K. Yadav, "Performance analysis of RSA and elliptic curve cryptography," *International Journal of Network Security*, vol. 20, no. 4, pp. 625–635, 2018.

[2] D. Mahto and D. K. Yadav, "RSA and ECC: A comparative analysis," *International Journal of Applied Engineering Research*, vol. 12, no. 19, pp. 9053–9061, 2017.

[3] D. Pittner, "Design and hardware implementation of an elliptic curve cryptography," M.S. thesis, Leopold-Franzens Univ. Innsbruck, Austria, 2022.

[4] N. J. G. Saho and E. C. Ezin, "Comparative study on the performance of elliptic curve cryptography algorithms with cryptography through RSA algorithm," in *Proc. CARI 2020*, ÉcolePolytechnique de Thiès, Senegal, 2020, pp. 1–10.

[5] M. R. Khan, K. Upreti, M. I. Alam, H. Khan, S. T. Siddiqui, M. Haque, and J. Parashar, "Analysis of elliptic curve cryptography & RSA," *Journal of ICT Standardization*, vol. 11, no. 4, pp. 355–378, 2023.

[6] K. L. Vasundhara, Y. V. S. S. Pragathi and Y. S. K. Vaideek, "A Comparative Study of RSA and ECC," *International Journal of Engineering Research and Application*, vol. 8, no. 1, pp. 49–52, Jan. 2018.

[7] M. Y. Mahardika and A. Triayudi, "Comparative analysis of performance of the encryption and decryption times of cryptography," *SAGA: Journal of Technology and Information Systems*, vol. 2, no. 3, pp. 304–310, Aug. 2024.

[8] M. Suárez-Albela, P. Fraga-Lamas, and T. M. Fernández-Caramés, "A practical evaluation on RSA and ECC-based cipher suites for IoT high-security energy-efficient fog and mist computing devices," *Sensors*, vol. 18, no. 11, Art.no. 3868, Nov. 2018.