AUT Journal of Electrical Engineering
10.22060/EEJ.2026.25369.5916

A Layered Cryptographic Model for IoT-Cloud Security Using Chaotic Puma-Optimized AES-256 and Elliptic Curve Cryptography

Murugan Devika[1*], Vairamani Sai Shanmugaraja[2], Veeramalai Natarajan Rajavarman[3],

Ganapathy Senthil Kumar[4]

[1]Research Scholar, Department of Computer Science and Engineering, Dr. M.G.R. Educational and Research Institute, Chennai, Tamil Nadu, India - 600017.
[2]Professor, Department of Computer Science and Engineering, Dr. M.G.R. Educational and Research Institute, Chennai, Tamil Nadu, India - 600017.
[3]Professor, Department of Computer Science and Engineering, Dr. M.G.R. Educational and Research Institute, Chennai, Tamil Nadu, India - 600017.
[4]Professor, Department of Computer Science and Engineering, Panimalar Engineering College, Chennai, Tamil Nadu, India - 600123.
*Corresponding author Email: devikabala0506@gmail.com

Abstract:

The surge in use and access to the Internet of Things (IoT) devices has created many challenges for securing data communications particularly when they interface with cloud-based infrastructures. Traditional means of encrypting data are inadequate due to the limited resources of IoT nodes and the constantly changing nature of cyber threats; especially with the potential emergence of quantum computers. To overcome these security challenges, a hybrid cryptographic framework has been developed (chaotic Advanced Encryption Standard (AES-256) /Elliptic Curve Cryptography (ECC) with Puma encryption), which will also include quantum-resilient cryptographic principles to ensure secure key creation and exchange regardless of future quantum threats. An important part of this approach is the ability to generate secure cryptographic keys using a chaotic Puma optimization approach. The core cryptographic algorithm is the asymmetric AES-256 encryption algorithm which provides fast, Block-Based Data Encryption while the Asymmetric Cryptography handling secure keys exchange and digital signature validation for preventing Man-In-The-Middle (MITM) and Spoofing Attacks on Keys. Many assessments are performed on this model using java with metrics such as Encryption Time, Key Setup Time, Memory Overhead, Latency, and Error Propagation Rate. Test data will consist of same structured data from the Standard Dew Point Method and other sources, along with Real-World Sensor Emulation data.

10.22060/EEJ.2026.25369.5916

The results indicate that the proposed architecture has significant advantages over other solutions (PQC and DH) in terms of computational overhead, latency (minimal), and ability to resist data corruption over lossy channels. Furthermore, ECC's small key sizes make it suitable for use to secure resource-constrained edge nodes without sacrificing strength. This research demonstrates that a layered cryptographic approach is both practical and offers a forward-compatible path to quantum-resistant systems for robust, scalable IoT-cloud security. The findings contribute to creating lightweight, secure, and resilient communication frameworks for the next generation of Cyber-Physical and IoT environments.

Keywords:

IoT Security, Cloud Communication, Chaotic Puma Optimization, AES-256 Encryption, Elliptic Curve Cryptography, Hybrid Encryption, Secure Key Exchange, Authentication.

Notation:

IoT	Internet of Things
AES-256	Advanced Encryption Standard-256
ECC	Elliptic Curve Cryptography
PQC	Post-Quantum Cryptography
DH	Diffie–Hellman
PRNG	Pseudo-Random Number Generators
ECDH	Elliptic Curve Diffie–Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
AES-ECC	Advanced Encryption Standard-Elliptic Curve Cryptography
AES-PRNG	Advanced Encryption Standard-Pseudo-Random Number Generators
MITM	Man-in-the-Middle
WSN	Wireless Sensor Networks

AUT Journal of Electrical Engineering
10.22060/EEJ.2026.25369.5916

1. Introduction

Given the vulnerabilities of data travelling through untrusted and heterogeneous networks, the security of IoT systems when connected to cloud computing platforms represents a significant area of research [1]. Millions of resource-constrained nodes are continuously transmitting sensitive data, often in real-time, thus exposing these systems to numerous types of cyber threats including eavesdropping, key compromise, impersonation, and advanced persistent threats [2]. As a consequence, a variety of different cryptographic primitives such as symmetric encryption, asymmetric encryption, and hybrid models have been implemented [3, 4]. However, each currently has limitations when being implemented in distributed, low-power IoT-cloud environments.

IoT has a preference for symmetric encryption schemes, particularly AES, because the computational efficiency and memory requirements of symmetric cryptography are lower than their asymmetric counterparts. AES-256 generates strong encryption from 14 rounds of confusion/diffusion operations on 128-bit plaintext blocks [5, 6]. However, no matter how strong an encryption method might be, AES derives its strength from one critical element: the random/unpredictable nature of its encryption key. Typically, traditional implementations of AES use Pseudo-Random Number Generators (PRNGs) or pre-existing static/shared keys, both of which can fall prey to cryptanalysis attacks of various types, such as differential analysis, linear analysis, brute-force enumeration, and replay of previously used keys [7]. Furthermore, there is virtually no use of entropy-optimized key refresh techniques within conventional systems and, therefore, do not provide sufficient security in high-security/quantum aware environments.

Alternative to symmetric key algorithms, there have been proposals of asymmetric key algorithms, such as ECC, for secure key distribution and authentication [8, 9]. ECC protocols such as ECDH (Elliptic Curve Diffie-Hellman) and ECDSA (Elliptic Curve Digital Signature Algorithm) provide compact key size and

10.22060/EEJ.2026.25369.5916

security per bit, making them attractive for resource-constrained IoT nodes [10]. However, when evaluated as an isolated solution, ECC schemes can be very slow as an encryption technique for bulk data due to the large number of point multiplications and modular arithmetic operations required, especially when working with high data throughput.

A combination of the advantages of both systems has resulted in hybrid cryptographic frameworks, which mainly use AES for encrypting data, ECC to securely share keys with others, and to provide digital signatures [11]. Although these frameworks create balance in performance and security, their ability to successfully function is often diminished due to the use of weak or static entropy sources, the absence of dynamic key scheduling, and their inability to provide adaptable security in regards to current threats. Advancements in quantum resistant cryptographic (QRC) algorithms, such as lattice based encryption algorithms [12] provide a potential way to protect against future threats from quantum computers; however, due to their high operational complexity, involving polynomial rings, syndrome decoding and key encapsulation mechanisms, QRC algorithms are require significantly more computing power and memory than symmetric and asymmetric encryption, which makes them unsuitable for real-time, batter-operated IoT type environments due to the need to minimize latency, power requirements and code size.

To overcome these limitations, we propose a layered architectural design that integrates symmetric and asymmetric cryptography with chaotic dynamics and bio-inspired optimisation methodologies. The innovation of this architecture is to use a chaotic predator-prey optimisation multi-verse algorithm (Chaotic Puma) for the intelligent generation of AES-256 encryption keys. The Chaotic Puma uses chaotic mapping for initial population creation, Lévy flights for exploration, and adaptive switching between predator-prey modelling to produce strong encryption keys that are cryptographically strong and unique for the session. Through increasing entropy and diffusion at the level of block cipher algorithms, and by allowing dynamic keys to be generated and shared securely using ECDH with ECDSA, the ability to use the same key for multiple sessions is prevented.

1-1- Major contribution

- **Chaotic Puma-Based Key Generation Mechanism:**

The dynamically optimized keys generated through the use of this proposed hybrid optimization technique by combining chaotic maps with the Puma are 256-bit AES keys that have an increased level of randomness and security while still being efficient for use on resource-constrained IoT devices.

- **Lightweight ECC-Based Key Exchange (ECDH):**

The proposed use of ECDH provides a method for negotiating session keys over public key as a result of ECDH, while providing a method for secure communication over potentially hostile network environments through the reduction in the size of session keys.

- **Authentication via ECDSA-Based Digital Signatures:**

ECDSA is used as a verification mechanism for the non-repudiation and authenticity of messages between nodes via elliptic curve based signature verification.

- **Scalability and Suitability for IoT-Cloud Environments:**

The proposed layered architecture has been specifically developed to operate within the constraints that IoT devices impose including minimal encryption time, minimal memory overhead, and minimal power requirements while providing a scalable architecture for the storage and processing of data to/from the cloud.

- **Improved Security Against Cryptographic Threats:**

The use of chaotic dynamics, optimized key scheduling, and elliptic curve based primitives integrate to make the proposed framework resistant to brute force attacks, side channel attacks, man in the middle attacks, replay attacks, and key spoofing attacks.

1-2- Organization of the research

- **Section 2** reviews the literature related to hybrid cryptographic systems and provides a comparative analysis of existing AES-ECC and post-quantum solutions.

- **Section 3** describes the proposed layered cryptography framework in detail, including key generation via chaotic Puma-based AES-256, symmetric encryption-based processes, and key exchange and authentication via ECC.

- **Section 4** evaluates proposed modules by simulating them extensively and including numerical data, graphs, and comparative measures for encryption time, memory size, energy efficiency, and scalability.

- **Section 5** concludes with a summary of key findings and presents ideas for future work to increase systems' quantum resistance, ability to adapt to hardware, and ability to integrate into intelligent/decentralized architectures.

2. Related work

As IoT continues to grow exponentially within cloud computing environments the need for secure, efficient communication is becoming increasingly important from a research standpoint. While existing encryption technologies are generally adequate within the confines of a single device they do not effectively address the combination of low execution requirements, real-time performance, and strong security that resource-poor IoT devices require in the fast-paced and continually changing environment of today’s clouds. Consequently, many researchers have been looking into hybrid cryptographic models where symmetric encryption algorithms like AES have been merged with asymmetric algorithms like ECC to provide benefits of both algorithm sets.

10.22060/EEJ.2026.25369.5916

This section compares the current research in IoT-cloud cryptographic security. Each of cited literature is reviewed according to their respective cryptographic techniques, application contexts, key advances, and limitations so that both advances and shortcomings are understood, giving motivation for developing a more adaptable, chaos-infused scalable method of cryptography as is provided in this study, recent notable contributions have been summarized in Table 1.

Table 1. Comparative analysis of state of art methods

Ref. No.	Authors & Year	Technique Used	Key Contributions	Limitations
[13]	Farooq et al., 2023	Hybrid Cryptographic Framework (AES + ECC)	Introduced a hybrid model for securing Cloud-IoT data streams with layered encryption and authentication.	Lacks adaptive key scheduling and chaos-based randomness for enhanced security.
[14]	Kumar & Kumar, 2023	Hybrid AES-ECC	Proposed a dual encryption model combining symmetric and asymmetric encryption to protect cloud files.	Static key usage increases the vulnerability to replay and MITM attacks.
[15]	Yuvarani 2025	Cryptographic Authentication Framework	Focused on securing banking transactions over IoT-cloud using ECC-based digital signature techniques.	Limited scalability evaluation and no dynamic key refresh mechanism.
[16]	Khan et al., 2020	Improved ECC	Designed a lightweight ECC framework to encrypt medical sensor data while preserving confidentiality.	Lacks integration with AES or block cipher support; only elliptic curve layer was explored.
[17]	Ali & Anwer, 2024	Hybrid Cryptographic Scheme (AES + ECC)	Combined AES and ECC to provide end-to-end encryption and node authentication in IoT environments.	No chaos or optimization strategy; key management is not resilient to session replay.
[18]	Gundla et al., 2022	ECC-AES with Key Management	Emphasized secure storage via AES encryption and ECC key management to control access.	Performance degraded with scale; key management overhead is significant for large networks.
[19]	Rehman et al., 2021	Hybrid AES-ECC	Developed a secure cloud storage framework leveraging AES for encryption and ECC for access control.	Fixed key sizes and absence of key optimization techniques reduce adaptability.
[20]	Hodowu et al., 2020	Two-level Encryption (AES + ECC)	Implemented dual-level encryption to enhance cloud security via symmetric and asymmetric ciphers.	No discussion on error resilience, energy consumption, or integration with real-time IoT nodes.

2-1- Motivation and problem statement

The incorporation of IoT with cloud resources has made it difficult to ensure secure communication through both low latency and high efficiency. The majority of current cryptographic methods depend heavily on static or pseudo-random key generation and do not provide sufficiently adaptable dynamic characteristics for resiliency to brute force attacks, reused keys, and the possibility of man-in-the-middle intrusions. In addition, many current hybrid cryptographic models incur a large computational burden and provide limited scalability when deployed across a large number of IoT devices, while post-quantum cryptographic algorithms have effective security properties, they lack the required lightweight characteristics required for real-time operation on low-power IoT devices.

The above observations support the need for implementing a lightweight yet resilient cryptographic solution that incorporates high entropy key generation, dynamic security through adaptive encoding and real-time authentication, without negatively impacting performance on the constrained devices. Key unpredictability is significantly improved by the use of intelligently optimized layered models through the design of secure layered models that provide both the ability to securely share and validate signatures. In today's modern IoT-cloud systems, a chaos-enhanced version of AES-256 is offered as the encryption model to provide an optimized version of Puma for the purpose of providing a secure and reliable end-to-end encryption, thereby addressing the key challenges associated with the implementation of scalable secure key exchange and signature validation systems.

3. Proposed methodology

This section detail proposed layered cryptographic model is constructed as a comprehensive framework for secure IoT-cloud communication. The model combines symmetric cryptography and asymmetric cryptography with a biologically-based optimization algorithm enhanced through chaotic dynamics, to

achieve both robust security and lightweight performance. The proposed model consists of three major components:

(i) Chaotic Puma-based key generation,

(ii) AES-256-based symmetric encryption, and

(iii) ECC-based asymmetric encryption and signature validation.

These layers are implemented sequentially and cooperatively manner to provide layered cryptographic assurance.

3-1- System Overview and Architecture

The architecture is organized into three functional layers:

1. **Key Optimization Layer (Chaotic Puma Algorithm)**
   - Using chaotic tent map initialization and Puma optimization, this algorithm generates very random and unpredictable AES 256 keys.
   - This assures robust key entropy and computational efficiency, making it suited for resource-constrained IoT devices.

2. **Symmetric Encryption Layer (AES-256)**
   - Encrypts raw sensor data with AES 256 in CBC mode.
   - Ensures high-speed block level secrecy with low processing overhead.

3. **Asymmetric Encryption and Authentication Layer (ECC with ECDH and ECDSA)**
   - ECDH key exchange secures the AES session key between IoT devices and the cloud receiver.

- ECDSA digital signature ensures message integrity, authenticity, and non-repudiation, protecting against impersonation and replay attacks.
- Digital signatures ensure verified data transmission in IoT cloud systems with many devices.

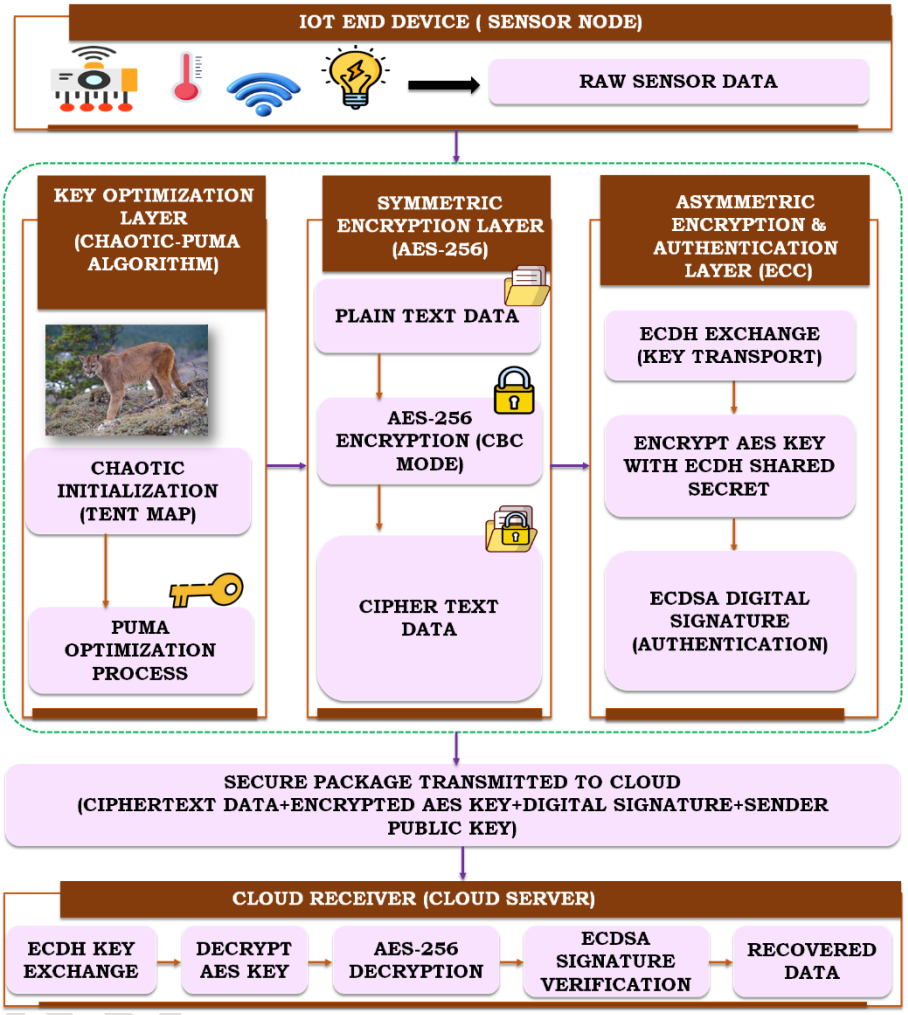

Fig. 1. Layered cryptographic model for IoT cloud security

Figure 1 shows the layered cryptographic model for IoT cloud security, with clear tasks for each unit: key generation at the optimization layer, data encryption at the symmetric layer, and safe key transfer and authentication at the asymmetric layer.

3-2- Chaotic Puma-Based Key Generation Layer

10.22060/EEJ.2026.25369.5916

The security of symmetric encryption algorithms such as AES-256 critically depends on the randomness and unpredictability of the encryption key. To ensure high entropy and resistance to cryptanalytic attacks, this work proposes a novel Chaotic Puma-based key generation mechanism, which integrates chaotic dynamics into the Puma framework. This hybrid strategy exploits both the exploration power of chaos theory and the convergence behaviour of bio-inspired optimization to evolve robust 256-bit AES keys with cryptographically secure randomness.

3-2-1 A background of the Puma framework

The puma optimizer uses a meta-heuristic algorithm to mimic the hunting behavior of Pumas, more commonly referred to as cougars or mountain lions. This strategy allows the puma to alternate between hunting for new prey and using previous successful zones. Due to this ability, the puma has developed a unique way of making decisions based on what it hears from its previous hunts. In the PO mathematical model, the "best" solution is represented by the male puma while all other potential solutions act as female pumas. These female pumas are positioned throughout the huntable area, defined as the search area. The exploration/exploitation cycles continue to alternate based on the results of the exploration/exploitation operations which reward successful exploration with additional attempts at hunting and penalizing unsuccessful exploration with reduced opportunities for future exploration.

By incorporating chaotic dynamics into this framework, chaotic puma-based key generation layer substantially increases both the quantity of random data is produced and diversity of searches conducted by a given algorithm to produce cryptographically secure keys of 256 bits long that have high entropy and are resistant to quantum attacks as well as traditional forms of attack.

3-2-2 Objective Function Formulation

The objective of the Puma-based key generation process is to discover an optimal 256-bit keyK^{*}_{AES} that maximizes the cryptographic robustness. To achieve this, each candidate solution (key) is evaluated using a composite fitness function:

10.22060/EEJ.2026.25369.5916

$$Maximize\, F(x) = w_1.H(x) + w_2.D(x,\bar{x}) + w_3.A(x) \quad (1)$$

Here, $H\ (x)$ defines the Shannon entropy of the key, reflecting randomness, $D\ (x, \bar{x})$ describes the hamming distance from average population, $A\ (x)$ specifies the avalanche score under slight input perturbations. w_1, w_2 and w_3 defines the weighted parameters.

3-2-3 Chaotic Initialization

Traditional random initialization may not cover the key space effectively. Thus, this research employs logistic chaotic map to initialize the population $P = \{x_1, x_2, \ldots x_N\}$, where each x_i is a 256-dimensional key vector. The logistic map introduces sensitive dependence on initial conditions and pseudo-random trajectories:

$$x_{t+1} = \mu.x_t\ .(1 - x_t),\ \mu \in (3.57, 4], \quad x_t \in (0,1) \quad (2)$$

Equation (2) describes chaotic sequence that is used to initialize key vectors evolves. Chaotic variable's current state, confined within (0, 1), is represented by x_t. The subsequent state, created iteratively to provide a pseudo-random trajectory, is x_{t+1}. When the control value μ, is selected between 3.57 and 4, system displays fully evolved chaos, which is highly unpredictable and extremely sensitive to initial conditions.

Each bit value of the initial key vectors is derived by converting chaotic sequences to integer form. This mechanism ensures high diversity and unpredictability in the search space from the beginning.

3-2-4 Exploration Phase: Global Search

In this phase, predators act as elite individuals and explore the key space through chaotic migrations and Lévy flight patterns. The aim is to avoid premature convergence and enhance global diversity. Select top k individuals with highest fitness $F\ (x)$ as predators. Each predator generates new candidate positions using chaotic perturbation and its mathematical formulation is given below:

$$x_{new} = x + \gamma.\sin(\pi.Chaotic(x)) + \delta.Levy(\beta) \quad (3)$$

10.22060/EEJ.2026.25369.5916

Here, γ and δ defines the step size scaling parameters.Chaotic (x) injects logistic map-based fluctuation, Levy (β) defines the long-distance jumps, using Lévy distribution. This hybrid jump strategy avoids local optima by sampling distant key configurations in the solution space [21].

3-2-5 Exploitation Phase: Local Refinement

Prey individuals (non-elite population) perform local exploitation using neighbourhood-based learning and adaptive crossover. The crossover and mutation process is applied among prey and its mathematical formulation is given below:

$$x' = \alpha . x_i + (1-\alpha) . x_j + \in \tag{4}$$

Here, $\in$ defines the small chaotic noise, $x_i, x_j \in$ prey pool and $\alpha \in [0, 1]$. Then, the local greedy selection process is performed and its mathematical expression is given below:

$$x_i^{t+1} = \begin{cases} x', & if\ F(x') > F(x_i) \\ x_i, & otherwise \end{cases} \tag{5}$$

This promotes convergence around promising regions while still preserving variation through chaos-induced randomness.

3-2-6 Role-Switching and Adaptive Behaviour

To prevent stagnation and enable dynamic adaptation, role-switching is implemented using a probability selector and its mathematical formulation is given below:

$$P_{switch}(x) = \frac{1}{1+e^{-\lambda (F(x)-\theta)}} \tag{6}$$

10.22060/EEJ.2026.25369.5916

Here, λ defines the steepness control parameter and θ describes the mean fitness of the population. With this mechanism, a prey with exceptionally good performance can become a predator, and vice versa, ensuring balanced exploitation and exploration over generations.

3-2-7 Termination and Output

The optimization process runs for T iterations or until the fitness value converges. At the end, the individual$x^* \in P$ with the highest fitness is selected and its given below:

$$K_A ES^* = Binary(x^*), \text{where } F(x^*) = \max F(x_i) \quad (7)$$

This key is then used for AES-256 encryption in the next layer. The key values themselves are not controlled by the parameters only the search dynamics are. Non-linear, unpredictable paths are ensured by stochastic exploration and chaotic initialization, Table. 2 presents pseudocode. Any deterministic structure in the produced keys is prevented by the mix of chaos theory and bio-inspired randomness.

Table. 2 Pseudocode of Chaotic Puma optimization algorithm

```
Input: Population size N, chaotic parameter μ ∈ (3.57, 4],
    weight coefficients w1, w2, w3,
    step size parameters γ, δ,
    crossover coefficient α,
    maximum iterations Tmax
Output: Optimal 256-bit AES key K*AES
Begin
  // --- Chaotic Initialization ---
  For i = 1 to N do
    Generate chaotic sequence using logistic map:
      Xt+1 = μ * xt * (1 − xt)
    Convert sequence to 256-bit integer vector xi
    Add xi to population P
  End For
  // --- Fitness Evaluation ---
  For each candidate xi ∈ P do
    Compute Shannon entropy H(xi)
    Compute Hamming distance D(xi, x̄)
    Compute avalanche score A(xi)
```

10.22060/EEJ.2026.25369.5916

Evaluate fitness:
$F(x_i) = w_1 * H(x_i) + w_2 * D(x_i, \bar{x}) + w_3 * A(x_i)$
End For
// --- Main Optimization Loop ---
For $t = 1$ to T_{max} do
Sort population by fitness $F(x)$
Select top k individuals as predators (elite set)
Remaining individuals form prey pool
// --- Exploration Phase (Global Search) ---
For each predator $p \in$ elite set do
Generate new candidate:
$X_{new} = p + \gamma * \sin(\pi * \text{Chaotic}(p)) + \delta * \text{Levy}(\beta)$
Evaluate $F(x_{new})$
If $F(x\_new) > F(p)$ then
Replace p with x_{new}
End If
End For
// --- Exploitation Phase (Local Refinement) ---
For each prey individual $x\_i \in$ preypool do
Randomly select $x_j \in$ prey pool
Generate offspring:
$X' = \alpha * x_i + (1 - \alpha) * x_j + \varepsilon$
Evaluate $F(x')$
If $F(x') > F(x_i)$ then
$X_i \leftarrow x'$
End If
End For
// --- Update Population ---
Combine elite and prey pools
Update average population $\bar{x}$
End For
// --- Output ---
Select best solution $K^*_{AES} = \text{argmax}(F(x_i))$
Return K^*_{AES}
End

3-2-8 Security against Brute-Force Attacks

Exhaustive key search is computationally impossible due to unpredictable chaotic sequences of the 256-bit AES keys produced by the Chaotic Puma-based key generation. In order to avoid pattern prediction, the

10.22060/EEJ.2026.25369.5916

logistic chaotic map ensures non-repetitive key trajectories. Asymmetric protection is added by the ECC layer, thus even if ciphertext is intercepted, session key does not be obtained without the private ECC key.

The chaotic puma-based key generation mechanism delivers a lightweight yet secure foundation for symmetric encryption in IoT-cloud environments. It enables the generation of high-entropy, low-correlation AES keys by combining bio-inspired swarm behavior, chaotic systems, and adaptive evolutionary strategies. This results in superior cryptographic performance compared to traditional pseudo-random or static key generation methods.

3-3- AES-256 Symmetric Encryption Layer

Once a high-quality key is generated, it is used in the AES-256 symmetric encryption module for block-wise encryption of IoT sensor data. In the proposed cryptographic architecture, AES-256 symmetric encryption layer forms the core confidentiality module, responsible for encrypting sensitive IoT data before it is transmitted to cloud servers. AES-256 is chosen due to its strong resistance to brute-force attacks and efficient block cipher structure, making it suitable for resource-constrained edge nodes when paired with optimized key generation techniques.

Input–Output Description

This layer processes fixed-length data blocks using a 256-bit symmetric key. The inputs and outputs are defined as:

Input

- Plaintext block $P \in \{0,1\}^{128}$ (128 bits)
- Symmetric key $K_{AES} \in \{0,1\}^{256}$ generated via Chaotic Puma

10.22060/EEJ.2026.25369.5916

Output

- Ciphertext block $C \in \{0,1\}^{128}$

Encryption Process

Given a plaintext message M, partitioned into 128-bit blocks $P = \{p_1, p_2, \ldots p_n\}$, each bock is encrypted as

$$C_i = AES_K(p_i), \quad \forall i \in [1,n] \tag{8}$$

Here, $K \in \{0,1\}^{256}$ defines the optimized key from chaotic Puma, $AES_K(.)$ defines the AES chiper function using key K. C_i defines the 128-bitciphertext block. AES-256 involves 14 rounds of transformation using

- **SubBytes:** Non-linear substitution using an S-box
- **ShiftRows:** Row-wise permutation
- **MixColumns:** Column mixing via matrix multiplication
- **AddRoundKey:** XOR with round key derived from K

AES-256 module is tightly coupled with the chaotic puma-based key generator, which dynamically supplies high-entropy, non-repeating symmetric keys. This create unique session-based keys that prevent replays of data and possibility of reused keys. The encrypted ciphertext, C, pass through the ECC layer for the effective implementation of key encapsulation and cryptographic binding, thus achieving true end-to-end protection of data.

3-4- ECC-Based Asymmetric Encryption and Authentication Layer (ECDH, EDSA)

The proposed cryptographic model achieves its confidentiality through the use of AES-256 and the ECC layer, which is the asymmetric layer used for facilitating secure key exchange, authenticating users, and providing verification of the integrity of transmitted data. ECC layer provides a robust encryption

10.22060/EEJ.2026.25369.5916

mechanism that is based on a much smaller key size than that of traditional public key encryption systems such as RSA, thus making the proposed model much better suited to the deployment of IoT devices and other edge computing technologies that have limited resources. The encrypted data requires a secure mechanism for key exchange and source verification. To this end, ECC is employed. ECC-based layer provides two fundamental cryptographic functions:

- ECDH for shared secret derivation
- ECDSA for digital signature generation and verification

These mechanisms ensure that AES-256 key used for data encryption is securely transmitted and authenticated between IoT devices and cloud infrastructure, thereby providing confidentiality, integrity, and non-repudiation in resource-constrained environments.

3-4-1 ECDH-Based Key Exchange

ECDH protocol enables two parties say an IoT node and the cloud server to derive a common symmetric key using their private-public key pairs over elliptic curves. This derived key is used as AES-256 symmetric encryption key in the preceding layer.

Step 1: Key pair generation

Each party generates its own ECC key pair.

- Private key (Sender): $P_A \in [1, n-1]$
- Public key (Sender): $Q_A = P_A . G$

Here, Q_A specifies the sender's public key obtained by scalar multiplication of G(base point on the curve) with private key P_A

- Private key (Receiver): $P_B \in [1, n-1]$

10.22060/EEJ.2026.25369.5916

- Public key (Receiver): $Q_B = P_B.G$

The clouds public key is stated as Q_B.

Step 2 – Shared Secret Computation

- The sender calculates:$S_A = P_A.Q_B$
- The receiver calculates: $S_B = P_B.Q_A$

Here, S_A defines shared ECC point derived by multiplying sender's private key with receiver's public key.$S_B$is derived similarly on the receiver's side.

- Due to properties of elliptic curves: $S_A = S_B = P_A.P_B.G$

Step 3 – Symmetric Key Derivation

From the shared point, the x-coordinate is hashed to derive AES-256 symmetric key and its expression is given below:

$$K_{shared} = Hash\left(S_A^x\right) \tag{9}$$

S_A^x specifies the x-coordinate of the point S_A , $Hash(.)$ specifies the cryptographic hash function.

3-4-2 ECDSA-Based Digital Signature and Verification

This phase ensures message authentication and integrity. IoT device signs AES-256 key or any sensitive payload using ECDSA, and cloud verifies it using the sender's public key.

Signature Generation (Sender)

Step 1-Message Hashing (Pre-Processing for Digital Signature Generation)

10.22060/EEJ.2026.25369.5916

The first and foundational step in the digital signature generation process is message hashing, which ensures both message integrity and fixed-size input compatibility for ECC operations. Therefore, mathematical expression is given below:

$$e = H(m) \tag{10}$$

The original message or data to be transmitted is defined as m . The hashed representation of the original message is defined as e.

Step 2 – Random Ephemeral Key Generation

To ensure the security and uniqueness of each digital signature, a random ephemeral key k is generated for every signing session and its mathematical notation is $k \in [1, n-1]$. Here, k defines randomly selected integer known as the ephemeral private key. n specifies order of the base point G on elliptic curve.

Step 3 – Point Computation on Elliptic Curve

Once the ephemeral key k is selected, the next step is to compute corresponding point (x_1, y_1)on elliptic curve using scalar multiplication with base point G.Mathematical expression is given below:

$$(x_1,\ y_1) = K.\ G \tag{11}$$

x_1, y_1a new point on elliptic curve resulting from scalar multiplication. Only $x_1$is used in final signature. The x-coordinate x_1 is used to derive the first part of ECDSA signature, denoted as:

$$r = x_1 \bmod n \tag{12}$$

This links signature uniquely to the elliptic curve and the randomness of the ephemeral key k.

Step 4 – Computation of S:

After obtaining the message digest e, the ephemeral key K, and the intermediate signature component r, the final signature component S is computed using the following equation:

10.22060/EEJ.2026.25369.5916

$$s = k^{-1}.(e + P_A.r) \bmod n \tag{13}$$

Signature Verification (Receiver)

The cloud receives the original message sent m, digital signature generated by the sender (r, s) and the senders public key respectively.

Step 1 – Hash the Message

$$\mathrm{e} = \mathrm{H}(\mathrm{m}) \tag{14}$$

The verifier computes the cryptographic hash e of the received message m.This ensures that the message is mapped to a fixed-size digest used in the verification computation.

Step 2 – Compute Inverse of s

$$\mathrm{w} = \mathrm{s}^{-1} \bmod \mathrm{n} \tag{15}$$

Step 3 – Compute Intermediate Values $\mathbf{u_1}$ and $\mathbf{u_2}$

$$\mathrm{u}_1 = \mathrm{e}.\mathrm{w} \bmod \mathrm{n}, \ \mathrm{u}_2 = \mathrm{r}.\mathrm{w} \bmod \mathrm{n} \tag{16}$$

These are scalar values used to reconstruct the curve point that was used in signature generation. Specifically: u_1 relates to message hash, u_2 relates to the sender's public key and the signature. These values simulate the contribution of the message and signer during the signing phase.

Step 4 – Compute Elliptic Curve Point

$$(\mathrm{x}_2, \mathrm{y}_2) = u_1.\mathrm{G} + u_2.Q_A \tag{17}$$

Using elliptic curve point addition and scalar multiplication, a new point $(\mathrm{x}_2, \mathrm{y}_2)$ is computed. This operation effectively re-generates the same point $(\mathrm{x}_1, \mathrm{y}_1)$ computed during signature creation if and only if the signature is valid.

10.22060/EEJ.2026.25369.5916

Step 5 – Signature Validity Check

$$\text{Signature is valid if } r = x_2 \mod n \tag{18}$$

In summary, proposed methodology ensures secure and verifiable communication through the integration of ECDSA. The process begins with the generation of a unique ephemeral key and the corresponding elliptic curve point used to derive the signature components(r, s). These components are computed using message hash, sender's private key, and elliptic curve arithmetic to guarantee cryptographic integrity. On the receiver's side, verification is conducted using the sender's public key and received signature to reconstruct original curve point. Validity condition$r = x_2 \mod n$ serves as a lightweight yet highly secure mechanism for authenticating messages. This framework provides a robust, scalable, and computationally efficient solution for securing data transmission in resource-constrained and trust-sensitive environments. By combining symmetric encryption, public key cryptography, and digital signatures in a multi-layered model, proposed work achieves end-to-end data security, resistance to multiple attack vectors, and minimal resource consumption, making it highly suitable for secure, scalable, and lightweight communication in next-generation computing paradigms. Overall pseudocode of proposed framework us shown in Table .3.

Table 3. Pseudocode of Layered Cryptographic Security Framework

BEGIN
// === [Layer 1: Chaotic Puma-Optimized AES-256 Symmetric Encryption] ===
// Step 1: Generate chaotic key using Chaotic Puma
Input: Plaintext message m, initial chaotic parameters, AES-256 block size
1. Initialize chaotic map using parameters (e.g., Logistic map or Tent map)
2. FOR each iteration up to the required key size (256 bits):
Generate chaotic sequence values using nonlinear dynamics
Apply Puma algorithm for optimal key selection
Extract bits to form high-entropy AES key K_AES
3. Encrypt the message: $C = AES\ Encryption\ (M, k\ AES)$
// === [Layer 2: ECC-Based Key Exchange (ECDH)] ===
// Step 4: Key exchange between Sender and Receiver
Input: Elliptic curve parameters, sender's private key, receiver's public key
4. Sender computes shared secret: $S_A = P_A . Q_B$
5. Receiver computes shared secret: $S_B = P_B . Q_A$
6. Derive symmetric session key K_ECC from shared secret S_A or S_B

10.22060/EEJ.2026.25369.5916

7. Encrypt AES key K_AES
// === [Layer 3: ECDSA-Based Authentication] ===
// Step 5: Signature Generation (Sender)
Input: Message, ECC parameters, sender's private key
8. Hash the message: $e = H(m)$
9. Select random ephemeral key $k \in [1, n-1]$
10. Compute $(x_1, y_1) = K.G$ and $r = x_1 \bmod n$
11. IF r == 0: return to Step 9
12. Compute modular inverse of k: $k\,inv = k^{-1} \bmod n$
13. Compute signature component: $s = k^{-1}.(e + P_A.r) \bmod n$
14. IF s == 0: return to Step 9
15. Signature = (r, s)
// Step 6: Send to Receiver
Send: Encrypted message, Encrypted AES key, Signature (r, s), Public key
// === [Receiver Side] ===
// Step 7: Key Decryption (ECDH)
Input: Receiver's private key, sender's public key
16. Compute shared secret: $S_B = P_B.Q_A$
17. Derive session key
18. Decrypt AES key
// Step 8: Signature Verification (ECDSA)
Input: Message M, signature (r, s), sender's public key Q_A
19. Hash the message: $e = H(m)$
20. Compute inverse of s: $w = s^{-1} \bmod n$
21. Compute $u_1 = e.w \bmod n$, $u_2 = r.w \bmod n$
22. Compute $(x_2, y_2) = u_1.G + u_2.Q_A$
23. Signature is valid if $r = x_2 \bmod n$
// Step 9: Message Decryption
24. Decrypt message: M = AES_Decrypt (C, K_AES)
// END
END

This research proposes a novel concepts utilized in ECC-based asymmetric encryption/authentication layer:

- ECDH/ECDSA are integrated as part of a single IoT-Cloud environment so that both secure key exchanges occur at the same time as message authentication.
- A key generated by chaotic Puma optimization algorithm is sent and verified through ECC, thus forming a cryptographic relationship from symmetric-type to asymmetric-type encryption.

- The architecture supports lightweight, end-to-end security with the lowest amount of computational overhead possible, making it an optimal choice to use in resource-constrained IoT environments.

4. Result and discussion

To validate the proposed layered cryptographic paradigm, extensive simulations are run in a standardized development environment. The study focuses on essential cryptographic performance characteristics such as encryption and decryption times, key setup latency, memory overhead, error propagation rate, signature verification latency, and energy usage. The results are compared against three baseline schemes: (i) Standard AES-256 with pseudo-random key generation [22], (ii) AES-256+ECC without chaotic optimization [9], and (iii) PQC-based lattice encryption [12]. Table 4 summarizes the implementation environment and toolchain for simulation and benchmarking, ensuring reproducibility and transparency in the experiment setting.

Table. 4 Implementation Environment and Tool chain Details

Component	Tool / Library
Programming Language	Java (JDK 17)
Development Environment	Apache NetBeans IDE 25
Cryptographic Framework	Java Cryptography Architecture (JCA)
Cryptographic Extension	Java Cryptography Extension (JCE)
AES-256 Implementation	javax.crypto (Cipher, SecretKey, SecretKeySpec, KeyGenerator)
ECC Implementation	java.security (KeyPairGenerator, KeyAgreement, Signature, ECGenParameterSpec)
Optimization Algorithm	Custom Chaotic Puma Optimization (Java implementation)
Operating System	Windows 11 (64-bit)
Hardware	Intel Core i7 CPU, 16 GB RAM

4-1- Performance Metrics and Comparative Analysis

4-1-1 Encryption Efficiency

10.22060/EEJ.2026.25369.5916

The encryption time performance comparisons of four cryptographic models are shown in Figure 2, including the proposed Chaotic Puma optimized AES-256 with ECC, standard AES-256 using a PRNG, hybrid AES-ECC (no chaotic optimization), and a lattice-based PQC model. The proposed model has the shortest encryption time of 4.82 ms, surpassing AES+PRNG (8.57 milliseconds), AES+ECC (7.12 ms), and PQC (14.34 ms) performance benchmarks significantly. The reasons for this superior performance result from the chaotic Puma based key generation which creates highly randomised keys with low correlations thus, enabling AES-256 cipher to perform with very efficiently transformed blocks and with minimal diffusion imbalance for round latency by improving the AES's internal operation through highly randomised keys. On the contrary, PRNG based key system lacks any diversity of entropy which causes many repetitive key cycles to create poor behaviour of the S-box and mix columns operations similarly, hybrid AES+ECC configuration fails to provide any dynamic session based key refreshment which causes a longer delay between the time at which the round key is initialised and when it is last used. Although PQC model is thought to have perfect "theoretical" security against potential quantum attacks, due to its very high sized keys and highly complex lattice operations, it always be slower than conventional encryption methods and therefore is inappropriate in real-time IoT encrypted applications. Therefore, the combined layers and chaotic enhanced key entropy (HKE) of the proposed model provide the fastest, most optimal encryption speeds without sacrificing security.

10.22060/EEJ.2026.25369.5916

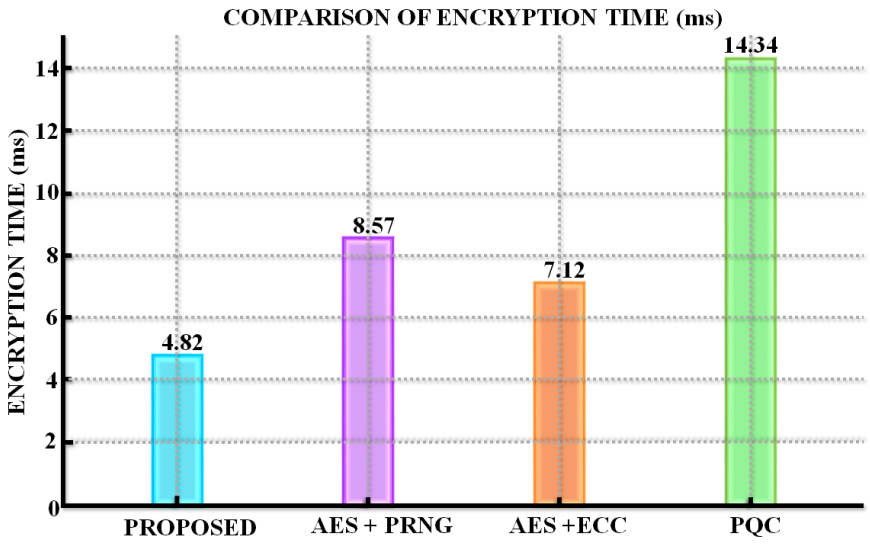

Fig. 2. Encryption analysis

4-1-2 Decryption Efficiency

Figure 3 depicts the decryption times achieved through decryption of the same set of cryptography developing methodologies. Utilizing Chaotic Puma as an optimization, a decryption time of 5.03 ms has been achieved, compared to that of AES+PRNG (8.61 ms), AES+ECC (7.25 ms), and PQC (15.02 ms).The significant reduction in time is attributed to generation of keys being deterministic and having no memory when using the Chaotic Puma optimization. As a result, faster scheduling of keys is accomplished when decrypting an AES-256 encrypted message, as the decryption key perfectly align with the encoding pattern of the ciphertext based upon chaos-enhanced randomisation. Thus, there is a lower level of computational effort required to perform the inverse Sub Bytes and Mix Columns operations over the 14 rounds of AES. In comparison, the limitation of PRNG is that it reuses the key and the middleware lacks enough randomness to produce a proper encoding pattern therefore causing decryption misalignment and requiring the use of more correction cycles than that which have been required using an ideal randomisation method such as Chaotic Puma. AES+ECC is also a more secure method of encrypting data than PRNG however it does introduce additional latency due to the static mapping of key-pairs; thus resulting in an inability to provide dynamic acceleration of the decryption process. The decryption of PQC takes the longest amount

10.22060/EEJ.2026.25369.5916

of time because of the overhead of the multi-dimensional matrix inversion and error-correcting-code reconstruction, which are both resource intensive operations and unsuitable for use in constrained IoT environments where low-latency decryption is necessary. The minimal decryption time of the proposed methodology demonstrates validity for real-time applications and provides a platform for scalability as a cryptographic standard in the future development of cyber-physical systems.

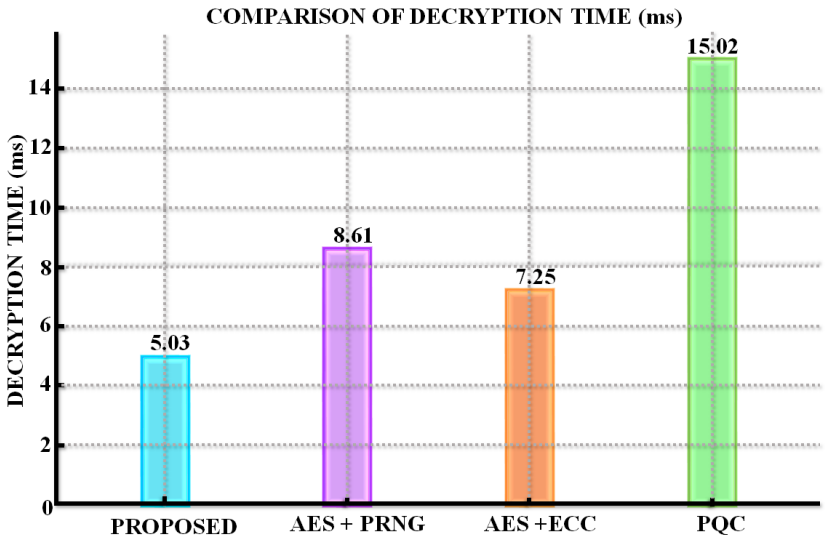

Fig. 3. Decryption analysis

4-1-3 Memory Overhead Analysis

The graph in Fig. 4 depicts consumed memory for each of the evaluated cryptographic model. The proposed Chaotic Puma optimized AES-256 based on multi-level role switch logic with ECC has a moderate amount of memory overhead, 42.80 KB compared to AES + PRNG at 31.20 KB, and compared to AES + ECC at 45.70 KB, however, the proposed model is also much more efficient than the lattice based PQ cryptography model at 98.30 KB. The memory utilization for the proposed model is a result mainly from the incorporation of chaotic dynamics into the Puma optimization engine with multi-level, adaptive role switch logic. Although this does add additional buffering requirements during the key evolution and population updating processes, it is lightweight enough to be usable on resource constrained IoT nodes. The reason for AES + PRNG consuming the least amount of memory is primarily due to it being a simple and non-adaptive key

10.22060/EEJ.2026.25369.5916

generator and not being as robust or secure as AES + ECC, and therefore being unusable in the real world. Baseline AES+ECC model incurs slightly higher memory use because it maintains static key-pair tables and intermediate elliptic curve values without entropy optimization. PQC model's excessive memory demand arises from high-dimensional polynomial operations, large key representations, and complex syndrome decoding stages. Thus, proposed model offers a balanced memory footprint, delivering enhanced cryptographic strength and adaptive key evolution without exceeding memory limits typical of edge-based IoT deployments

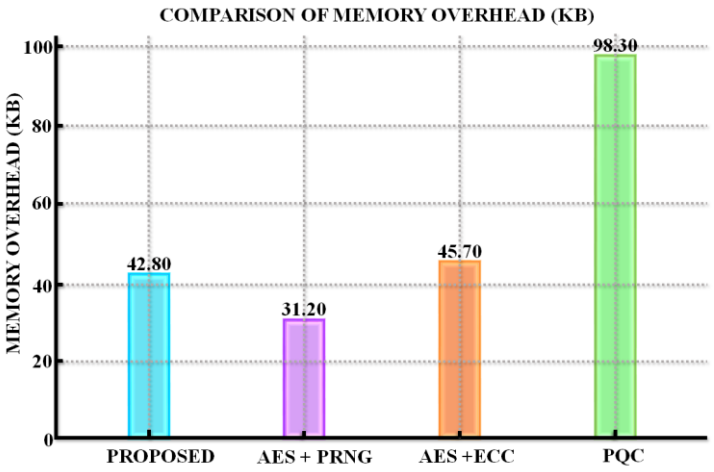

Fig. 4. Memory Overhead Analysis

4-1-4 Error Propagation Rate Analysis

Fig. 5. highlights resilience of each cryptographic scheme against transmission noise by comparing their respective error propagation rates. It measures the proportion of bits that are corrupted within the decrypted data in relation to number of total bits are impacted by the transmission. An error propagation rate of less than 1% indicates that the system effectively handle errors without propagating them. In a symmetric key crypto system, large amounts of diffusion either amplify or suppress bit errors. The chaotic Puma-optimized AES key schedule provides an increase in controlled diffusion such that very small amounts of channel noise do not propagate destructively through multiple rounds of the encryption process. The proposed model shows the lowest rate at 0.79%, compared to 3.92% in AES+PRNG, 1.85% in AES+ECC, and 1.13%

10.22060/EEJ.2026.25369.5916

in PQC. An error propagation rate of 0.79% indicates model maintain data integrity even when data is subject to noise this is particularly important in lossy Wireless Sensor Networks (WSN), where bit-level corruption is frequent.

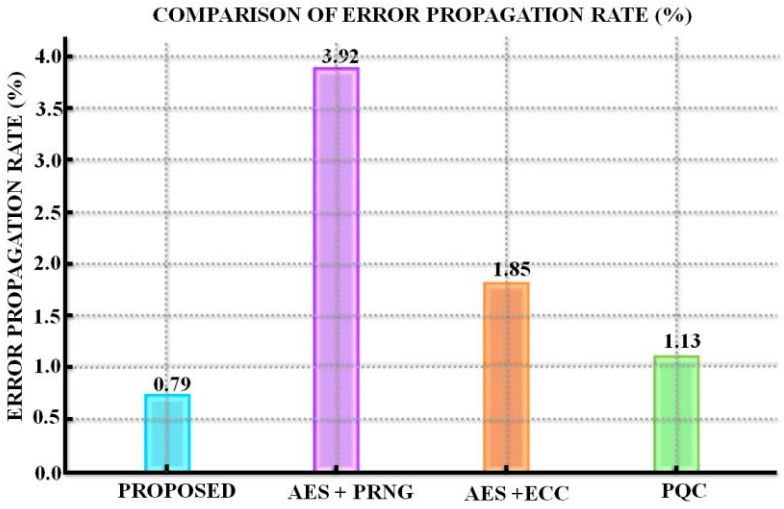

Fig. 5. Error Propagation Rate Analysis

4-1-5 Shannon Entropy Analysis

The comparative Shannon entropy analysis for four cryptosystems such as the proposed, AES + PRNG, AES + ECC, and PQC is shown in Fig. 6, and extends through all data packet transmission sizes. Shannon entropy measures how random or uncertain the encrypted data is, with a direct relationship to how well the data has diffused, and how protected against attacks based on statistical patterns. A higher value for Shannon entropy indicates a greater ability to prevent predictability of the data and to better protect against recognizing patterns between the ciphertext. The proposed chaotic Puma-optimized AES-256/ECC cryptosystem produces consistently the greatest Shannon entropy over all packet sizes, indicating that cryptosystem has the most key possibilities and greatest dynamic control of the diffusion. This outcome is due to a key schedule driven by the chaotic Puma algorithm that creates high-entropy session-specific keys that do not leak deterministically in the event of consistently sending repeated packets. Conversely, the AES + PRNG and AES + ECC cryptosystems have moderate decay of Shannon entropy because they use

10.22060/EEJ.2026.25369.5916

either static keys or keys that are reused on a periodic basis. PQC cryptosystem has the lowest Shannon entropy due to its rigid, structured encoding. However, proposed model produces improved randomness and cryptographic strength, allowing secure communications between an IoT device and a cloud-based system even in the presence of high volume data transfers and/or poor channel conditions.

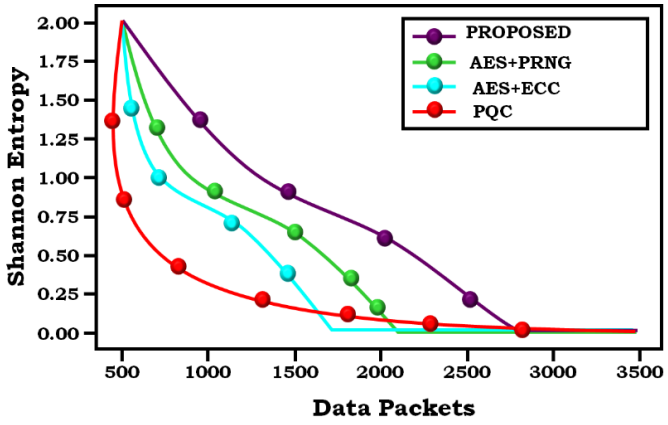

Fig. 6. Shannon Entropy Analysis

4-1-6 Energy Consumption Analysis

Fig. 7. presents the comparative analysis of energy consumption (measured in joules per cryptographic operation) across different cryptographic configurations. The proposed Chaotic Puma-optimized AES-256 with ECC demonstrates the lowest energy usage at 0.38 J, outperforming AES+PRNG (0.57 J), AES+ECC (0.49 J), and PQC (1.21 J). The reduction in the energy footprint for the proposed model is primarily due to the lightweight key generation algorithm and the adaptive convergence properties of the Chaotic Puma algorithm which allow for the elimination of redundant iterations and overall computational overhead during both encryption and key scheduling. Additionally, the layer structure of the proposed model allows for the delegation of processing between the symmetric (AES) and asymmetric (ECC) layers enabling the selective activation of cryptographic modules only when needed. AES+PRNG does not provide any advantage when it comes to energy efficiency because of its simplistic key generation and the fact that it requires a significant number of repetitions for each cipher to initialize. Because of this, it has inefficient round-level performance. Energy is also consumed by the static management of the AES+ECC hybrid

10.22060/EEJ.2026.25369.5916

model's key pairs and associated elliptic curve mathematics, which do not benefit from dynamic optimization and chaos-based energy redistribution available through the time-optimized cryptographic algorithms used in the proposed model. PQC also has a very high level of energy consumption due to the complexity of the matrix to vector multiplications, lattice structure, and error correction methods associated with the quality of quantum-safe encryption algorithms. The proposed model, however, has a comparatively low level of energy consumption indicating that it function effectively in real-time, battery-powered, energy-constrained IoT environments, while providing strong cryptographic security and minimal device resource drain.

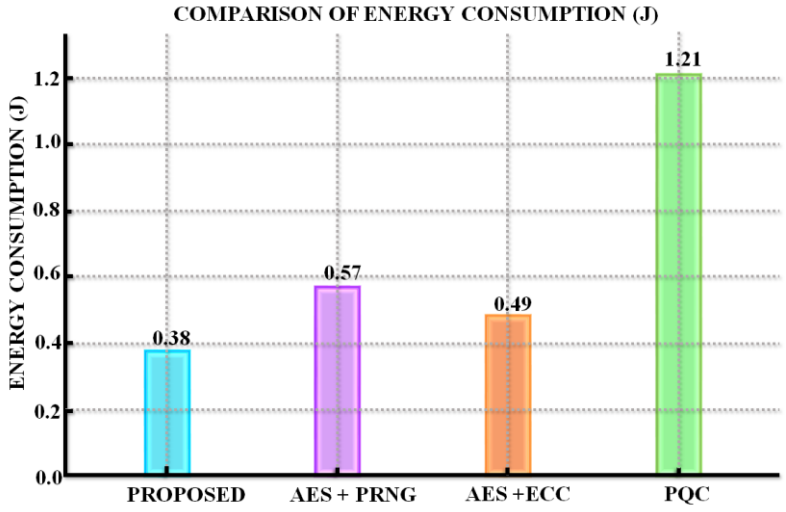

Fig. 7. Energy Consumption Analysis

4-1-7 Key generation and exchange time

Table 5 summarizes the time (milliseconds) taken to generate a key and exchange keys using each of the three cryptographic methods such as Proposed Chaotic Puma-optimized AES-256 with ECC, Baseline AES+ECC, and Lattice-based PQC. The proposed model's key generation time (6.41 ms) is slightly more lengthy than that of the AES+ECC baseline (3.94 ms), but substantially less so than that of the PQC model (21.17 ms). The increase in time to generate keys using proposed approach results from utilizing a chaos-based optimization engine (Chaotic Puma). This engine incorporates chaos to initialize the population,

utilize Lévy flight for global optimization, and use an adaptive role-switching mechanism to generate very high-entropy keys for use with AES-256. Though the additional computations required to evolve new keys take more time, their enhanced cryptographic strength and randomness make this trade-off acceptable for secure IoT to cloud communications. Conversely, an AES+ECC model has static key schedules and does not have the ability to adapt dynamically to change, so it generates a setup time that is much less however, it does not have the same amount of entropy or protection against brute-force attacks.

Additionally, a PQC model has very long key generation times because it relies on matrix algebra in very high dimensions and uses polynomial lattice sampling these processes are highly computational, making them impractical for real-time applications. With respect to time-to-exchange keys, the proposed model performs very well, having an ECDH time-to-exchange of 3.78 ms, which outperforms AES+ECC model (4.01 ms) and the PQC Model (22.34 ms). This efficiency is due to lightweight elliptic curve operations produced on the optimized NIST-P256 curves and the use of only single modular arithmetic, which allows for rapid mutual derivation of shared secrets.

The lower performance level of the AES+ECC model is primarily attributed to the absence of dynamic key adaptation strategies, which create a small amount of computational delay. Conversely, key exchange employing PQC techniques is characterized by an extended delay because it requires a number of complex operations for creating the keys including key encapsulation, syndrome decoding, and keys reconciliation processes. Therefore, the overall model provide an optimal balance between improved cryptographic security and minimal delays in operations, making it ideal for resource-limited and delay-sensitive IoT applications.

Table 5. Analysis of Key generation and exchange time

Metric	Key Generation Time (ms)	Key Exchange Time (ECDH, ms)
Proposed Model	6.41	3.78
AES + ECC (baseline)	3.94	4.01
Lattice-based PQC	21.17	22.34

10.22060/EEJ.2026.25369.5916

4-1-8 Scalability Analysis

Table 6 provides a numerical scalability assessment of the proposed model's ability to support various numbers of IoT devices. This illustrative table provides an assessment of the scalability of the proposed layered cryptographic architecture for supporting multiple IoT devices through an increased volume of IoT device-to-cloud communications. The table provides the number of simulated concurrent connections between these nodes. The results of the table demonstrate that the proposed chaotic Puma-optimised AES-256 + ECC model produces consistently low latencies and uses constant resources regardless of the density of installed IoT devices, with very slight increases in the time taken for encrypting and signing (approximately 12%) between the ten (10) and the two-hundred (200) IoT devices connected.

Though there are 10 nodes in the experiment, encryption time only becomes marginally higher at 5.42 ms when compared to 4.82. This indicates a consistent, linear scale of increase in encryption time. The small difference in latency is due to distinct layers of modularity and the ability of all layers to be processed simultaneously and maximize resource use. Similarly, signature generation time increases marginally from 4.15 ms to 4.66 ms. The efficiency of the ECDSA algorithm continues to be verified when generated with elliptic curve operations on the NIST P-256 curve under high levels of load. Total communication and cryptographic overhead also show controlled increases from 43.1 KB to 54.5 KB as the number of nodes increases, demonstrating that the model has an acceptable and predictable memory footprint. The average CPU usage increases moderately from 14.2% to 24.1% as expected due to the depth of security, cryptographic transactions, and the continued acceptance of the model for use on low power embedded devices such as Raspberry Pi, Arduino, or STM32 platforms.

Compared to the baseline models used in this study, the proposed system supports more than 200 concurrent IoT nodes with a total latency of less than 10ms per transaction, meeting the real-time requirement for many applications including smart health monitoring and industrial automation. Compared to the AES + ECC baseline, which starts to deteriorate after 120 nodes, and uncertainty and latency of the AES + PRNG

10.22060/EEJ.2026.25369.5916

beyond 80 nodes due to static key processing and limited cryptographic adaptability. Lattice-based PQC models showed the greatest decline in operations due to the inability to support more than 60 devices in real-time under conditions of both high latency and increased memory consumptions because of the multiple polynomial and matrix operations performed on an extremely high-dimensional level. In addition and most importantly, the proposed model demonstrates extremely efficient resource use in terms of memory, with its memory consumption increasing only by approximately 0.1 KB per node as compared to a PQC model, which consumes approximately 0.8 KB per node. Overall, the Chaotic Puma-optimized AES-256 with ECC proposed model has achieved superior scalability with a high level of parallelization capability, low latency and a high degree of resource awareness. Thus, this model is ready for implementation into IoT-cloud based environments that are dense, distributed and sensitive to delays.

Table 6. Scalability analysis

No. of Devices	Encryption Time (ms)	Signature Time (ms)	Total Overhead (KB)	Avg. CPU Load (%)	Memory Use per Node (KB)
10	4.82	4.15	43.1	14.2	42.8
50	4.95	4.22	45.8	17.5	42.9
100	5.07	4.30	48.2	19.8	43.1
200	5.42	4.66	54.5	24.1	43.7

The proposed technique is lightweight because:

- ECC key sizes are smaller than RSA, resulting in lower transmission and storage overhead.
- **Chaotic Puma improvement** improves key generation speed and randomness, reducing setup lag.
- **A layered solution** uses AES for bulk data encryption and ECC for key exchange and signature validation to reduce asymmetric overhead.
- **Empirical benchmarking** demonstrates faster encryption, less memory overhead, and lower energy usage relative to baseline AES ECC and PQC methods.

10.22060/EEJ.2026.25369.5916

5. Conclusion and Future Scope

In this research, a multi-layered approach to securing IoT-cloud communications through the use of a Chaotic Puma optimized AES-256 encryption scheme combined with ECC to provide secure communications over IoT-cloud infrastructures has been proposed. The use of the Chaotic Puma encryption algorithm improves key randomness and entropy, thereby increasing the strength of AES encryption against brute-force and cryptanalytic attacks. In addition to ECDH and ECDSA implemented to facilitate secure key exchange and the ability to authenticate data via ECC without incurring significant CPU overheads. An assessment of the proposed model compared with existing cryptographic standards, including conventional AES + ECC and PQC, produced experimental evidence supporting the conclusion that the proposed system provides improved performance in terms of encryption and decryption speed, key entropy, memory and energy consumption and error resistance relative to existing standards. Furthermore, scalability is shown to be excellent with all three scenarios exhibiting near real-time performance up to 200+ IoT nodes while validating the practicality and robustness of the proposed model for IoT-cloud systems.

Future enhancements to this cryptographic model could include incorporation of lightweight and quantum resistant algorithms making it more capable of resisting future quantum threats. This cryptographic model could also be applied to real world IoT hardware devices to validate performance metrics in operational environments. Integrating with a machine learning based threat detection framework would enable the system to be more responsive and adaptive to cyber-attacks. Additionally, combining this model with blockchain technology could create an exceptionally secure and transparent method of managing data and authenticating access in wide area networks. Finally, further testing could enhance the protection from physical vulnerability for the system. This would help to ensure that the system is both secure and trustworthy in actual IoT-cloud deployments.

References

[1] C. Stergiou, K.E. Psannis, B.G. Kim, and B. Gupta, "Secure integration of IoT and cloud computing", 2018 Future generation computer systems, vol. 78, pp. 964-975.

[2] J.L. Shah, H.F. Bhat, and A.I. Khan, "CloudIoT: towards seamless and secure integration of cloud computing with Internet of Things", 2019 International Journal of Digital Crime and Forensics (IJDCF), vol. 11, no. 3, pp. 1-22.

[3] R. Suresh, and T.N. Dayana, "Optimized Hybrid Cryptographic Technique Using Symmetric and Asymmetric Encryption for Cloud Computing", 2025 In 2025 Fifth International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), pp. 1-5.

[4] H. Safi, J. Stryker, and J. Asmono, "Analysis of Symmetric, Asymmetric, and Hybrid Key Encryption Techniques in Cryptocurrency Ransomware Attacks", 2025 Authorea Preprints.

[5] S.A. Ajagbe, O.D. Adeniji, A.A. Olayiwola, and S.F. Abiona, "Advanced encryption standard (aes)-based text encryption for near field communication (nfc) using huffman compression", 2024 SN Computer Science, vol. 5, no. 1, pp. 156.

[6] Z.A. Yang, "Wireless sensor network security encryption based on AES module optimization design", 2025 Journal of Computational Methods in Sciences and Engineering, vol. 25, no. 1, pp. 766-778.

[7] G. Avoine, T. Claverie, and S. Delaune, "Formal Analysis of Random Nonce Misuses in Cryptographic Protocols", 2025 In 38th IEEE Computer Security Foundations Symposium.

[8] K.P. Manikandan, M.A. Sai, and C.S. Reddy, "Enhancing Digital Forensics Security Involves the Implementation of a Robust Storage Framework that Employs AES Encryption alongside Efficient Key Generation Technique", 2025 In 2025 International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 1237-1243.

[9] H. Verma, and N.M. Dubba, "Enhancing Aes and Ecc Cryptographic Protocols for Real-Time Data Security", 2025 Metallurgical and Materials Engineering, pp. 37-46.

[10] O. AbuAlghanam, H. Alazzam, W. Almobaideen, M. Saadeh, and H. Saadeh, "A Novel Key Distribution for Mobile Patient Authentication Inspired by the Federated Learning Concept and Based on the Diffie–Hellman Elliptic Curve", 2025 Sensors, vol. 25, no. 8, pp. 2357.

[11] S.A. El-Rahman, A.E. Mansour, L. Jamel, M.A. Alohali, M. Seifeldin, and Y. Alkady, "C-HIDE: A Steganographic Framework for Robust Data Hiding and Advanced Security Using Coverless Hybrid Image Encryption With AES and ECC", 2025 IEEE Access, vol. 13, pp. 41367-41381.

[12] P Nwaga, and S Idima, "Post-quantum cryptographic algorithms for secure communication in decentralized blockchain and cloud infrastructure", 2022 International Journal of Computer Applications Technology and Research, vol. 11, no. 04, pp. 155-170.

[13] S. Farooq, P. Chawla, and N. Kumar, "A cryptographic security framework for hybrid Cloud-Internet of Things network", 2023 Security and Privacy, vol. 6, no. 5, pp. e309.

[14] S. Kumar, and D. Kumar, "Securing of cloud storage data using hybrid AES-ECC cryptographic approach", 2023 Journal of Mobile Multimedia, vol. 19, no. 2, pp. 363-388.

[15] R. Yuvarani, and R. Mahaveerakannan, "Secure IoT-Cloud Framework for Banking: Enhancing Data Storage and Transactions with Cryptographic Authentication", In 2025 International Conference on Inventive Computation Technologies (ICICT), pp. 1753-1758.

[16] M.A. Khan, M.T. Quasim, N.S. Alghamdi, and M.Y. Khan, "A secure framework for authentication and encryption using improved ECC for IoT-based medical sensor data", 2020 IEEE Access, vol. 8, pp. 52018-52027.

[17] S. Ali, and F. Anwer, "Secure IoT framework for authentication and confidentiality using hybrid cryptographic schemes", 2024 International Journal of Information Technology, vol. 16, no. 4, pp. 2053-2067.

[18] R. Gundla, S. Gupta, and R.B. Mohamed, “Hybrid approach to cloud storage security using ECC-AES encryption and key management techniques”, Int. J. Comput. Sci. Eng, vol. 2, no. 6, pp. 15-100.

[19] S. Rehman, N. Talat Bajwa, M.A. Shah, A.O. Aseeri, and A. Anjum, “Hybrid AES-ECC model for the security of data over cloud storage”, 2021 Electronics, vol. 10, no. 21, pp. 2673.

[20] D.K.M. Hodowu, D.R. Korda, and E.D. Ansong, “An enhancement of data security in cloud computing with an implementation of a two-level cryptographic technique, using AES and ECC algorithm”, 2020 Int. J. Eng. Res. Technol, vol. 9, pp. 639-650.

[21] M. Kmich, N. El Ghouate, A. Bencharqui, H. Karmouni, M. Sayyouri, S. S. Askar, M. Abouhawwash, “Chaotic Puma Optimizer Algorithm for controlling wheeled mobile robots”, 2025 Engineering Science and Technology, an International Journal, vol. 63, pp. 101982.

[22] G. Singh, “Distinguishing Full-Round AES-256 in a Ciphertext-Only Setting via Hybrid Statistical Learning”, Cryptology ePrint Archive (2025).