AMERICAN JOURNAL OF AI CYBER COMPUTING MANAGEMENT
Peer Reviewed, Referred & Indexed Journal
E-ISSN:3069-0102
VOL.6, NO. 2(2026)

# A Resource-Aware Authenticity Model for Secure Propagation of System-Level Updates in IoT Ecosystems

V. Sowjanya[1*], Datla Shravya[2], Jeripothu Sathwik[2], Kundella Yashwanth[2]

[1]Assistant Professor, [2]UG Student, [1,2]Department of Computer Science and Engineering

[1,2]Kommuri Pratap Reddy Institute of Technology, Ghanpur, Ghatkesar, 501301, Telangana, India.

*Correspondence: V. Sowjanya (sowjanya.vuriti89@gmail.com)

## ABSTRACT

The rapid progression of the Internet of Things (IoT) has significantly transformed how interconnected devices operate, exchange information, and enable automation across various domains. Although introduced in the late 20th century, IoT builds upon earlier advancements such as embedded systems, wireless communication, and standardized networking protocols. In its early stages, IoT systems relied heavily on centralized infrastructures for managing operations, particularly software updates, due to limited device capabilities. Traditional update mechanisms, including centralized servers and Over-The-Air (OTA) approaches, introduced critical challenges such as single points of failure, vulnerability to cyber threats, poor scalability, and lack of transparency in verification processes. As IoT ecosystems continue to expand, these issues increasingly affect system reliability and data integrity, highlighting the need for a decentralized and secure update management approach. To address these limitations, the proposed system integrates blockchain technology with Inter-Planetary File System (IPFS) to establish a distributed, tamper-resistant framework for managing updates. Ethereum-based smart contracts are used to handle user registration, maintain update records, and process transactions in a transparent and immutable manner. For enhanced security, the system utilizes Elliptic Curve Integrated Encryption Scheme (ECIES) combined with Ciphertext-Policy Attribute-Based Encryption (CP-ABE) to safeguard update content, which is then stored in IPFS using content-addressed storage for decentralized access. A Django-based web interface enables smooth interaction for users, while a Tkinter-based simulation environment models IoT device behavior and update dissemination guided by blockchain data. By removing centralized dependencies and incorporating secure, verifiable processes, the system improves reliability, trust, and scalability, aligning with modern trends in decentralized IoT architectures.

**Keywords:** Blockchain Technology, Over-The-Air (OTA) Updates, Data Integrity, Ciphertext-Policy Attribute-Based Encryption (CP-ABE), Elliptic Curve Integrated Encryption Scheme (ECIES), Inter-Planetary File System (IPFS)

## 1.INTRODUCTION

The rapid expansion of the Internet of Things (IoT) as shown in Fig. 1 has significantly accelerated the adoption of smart and interconnected devices across various domains due to their convenience, efficiency, and ability to automate tasks in everyday life. Over the past few years, IoT has emerged as a dominant paradigm for low-power and resource-constrained networks, enabling seamless communication among devices with limited computational and energy capabilities [1]. However, this exponential growth has also introduced serious security concerns, as IoT devices have increasingly become attractive targets for cyberattacks and malicious activities. Several studies, including the work have identified that many of these vulnerabilities stem from inadequate device management practices. Common issues include insecure network services, weak or absent update mechanisms, reliance on outdated or vulnerable software components, insufficient privacy protection measures, insecure data transmission and storage techniques, ineffective device management protocols, and unsafe default

configurations [4]. These shortcomings, whether originating from manufacturers or end users, can lead to severe consequences such as unauthorized access, data breaches, device malfunction, and large-scale network exploitation [2].

To address these risks, maintaining up-to-date firmware becomes a critical requirement for ensuring the security and proper functioning of IoT devices. Firmware refers to the embedded set of instructions and programmed routines within a device that control its hardware components and operational behavior. Regular firmware updates help patch vulnerabilities, improve performance, and introduce new functionalities. Traditionally, these updates are delivered by device manufacturers through Firmware Over-The-Air (FOTA) mechanisms, where updates are transmitted via the internet to deployed devices. The FOTA approach typically relies on a centralized client-server architecture, in which a Firmware Management Center (FMC) acts as the central server responsible for storing and distributing firmware updates, while IoT devices function as clients that receive and install these updates [3].

Despite its widespread use, the centralized FOTA model suffers from several critical limitations, particularly in large-scale IoT deployments. As the number of connected devices increases dramatically, the FMC can become a performance bottleneck, struggling to distribute firmware updates efficiently to all devices within a reasonable timeframe. This issue is further exacerbated when multiple devices attempt to download updates simultaneously, leading to network congestion and degraded system performance. Additionally, the centralized nature of the FMC introduces a single point of failure, making the entire update process vulnerable to system outages, targeted cyberattacks, or server compromises. Another major limitation is the requirement for continuous and stable internet connectivity for each IoT device, which is not always feasible in remote or resource-constrained environments [4].

These challenges highlight the urgent need for a more robust, scalable, and secure firmware update framework that overcomes the drawbacks of centralized architectures. Such a framework should be capable of distributing updates efficiently across large and heterogeneous IoT networks, minimizing dependency on constant internet connectivity while ensuring data integrity, authenticity, and trust. Moving towards decentralized approaches can address these issues by eliminating single points of failure, improving scalability, and enhancing the overall security of firmware distribution processes in modern IoT ecosystems [5].

## 2. RELATED WORK

### 2.1 Blockchain-Based Identity, Access Control, and Trust Mechanisms in IoT

Khayer, et al. [6] presented a comprehensive survey on blockchain integration in IoT systems, focusing on identity management, access control, and trust mechanisms. Their study highlighted that decentralized identity models and blockchain-based authentication significantly enhance security by eliminating reliance on centralized authorities and reducing single points of failure. The authors examined smart contract-based access control mechanisms that automate authorization and enforce fine-grained policies in distributed environments. Additionally, they explored trust management approaches leveraging blockchain's immutability and transparency to ensure reliable interactions among devices, while also identifying challenges such as scalability, computational overhead, interoperability, and privacy concerns.

### 2.2 Lightweight and Scalable Blockchain Architectures for IoT

Xu, et al. [7] proposed a blockchain framework tailored for Industrial IoT environments with a focus on efficiency in resource-constrained settings. Their layered architecture separates low-capability

devices from more powerful nodes, enabling optimized workload distribution. By redesigning block structures and introducing a lightweight consensus mechanism, the framework reduces overhead and improves performance. Similarly, Na, et al. [8] introduced a federated blockchain architecture that clusters nodes into groups managed by leader nodes, reducing network congestion and improving scalability. Their approach enhances latency and fault tolerance by enabling localized consensus and distributed data handling, making it suitable for large-scale deployments.

### 2.3 Secure Transaction Management and System Integration

Basudan, et al. [9] presented a scalable blockchain framework integrating cloud and fog computing to manage secure transactions in dynamic IoT environments. Their system employs certificate-based authentication and role-based access control to ensure secure interactions, while smart contracts automate transaction validation and data exchange. The framework supports heterogeneous devices and dynamic participation, ensuring data integrity and confidentiality. However, the authors noted increased system complexity and the need for efficient resource management as key challenges.

### 2.4 Trust Management and AI-Integrated Blockchain Approaches

D'Aniello, et al. [10] explored trust management in IoT systems by combining blockchain with AI-based techniques. Their study highlighted that blockchain provides secure and tamper-resistant data handling, while AI enables dynamic and context-aware trust evaluation. They analyzed multiple trust computation models and identified limitations such as scalability issues, privacy concerns, and computational complexity. The work emphasizes the need for adaptive and intelligent trust frameworks to address evolving IoT requirements.

### 2.5 Blockchain–IoT Integration and Performance Optimization

Delladetsimas, et al. [11] reviewed blockchain–IoT integration with a focus on system architectures and practical applications. Their study analyzed various blockchain models and consensus mechanisms, evaluating their impact on scalability, energy consumption, and latency. They also discussed the role of supporting technologies such as edge computing and decentralized storage in improving system efficiency. Furthermore, Maftei, et al. [12] proposed a scalable framework using gateway-based data aggregation, where multiple data packets are combined into a single transaction. This approach significantly reduces bandwidth usage, network congestion, and latency, improving overall system performance in high-density IoT environments.

### 2.6 Blockchain-Based Secure Firmware Update Mechanisms

Yohan, et al. [13] proposed a blockchain-based firmware update framework supporting both direct and peer-to-peer update models. Their approach enhances flexibility and availability in distributed IoT environments by reducing dependence on centralized systems. While the design improves efficiency in update distribution, the study identified a critical limitation in message verification, which may expose the system to security threats such as data tampering and malicious injection.

## 3. PROPOSED SYSTEM

The system represents a secure and scalable IoT software update distribution framework that integrates encryption, decentralized storage, and blockchain-based transaction management to ensure reliable and tamper-resistant update delivery, as illustrated in Fig. 1. The process begins with the Manufacturer interacting with a Django-based web application, which manages user authentication, registration of manufacturers and IoT owners, firmware upload operations, and secure transaction handling. Once a firmware file is uploaded, it is divided into multiple smaller blocks to optimize transmission efficiency and parallel processing. These blocks are encrypted using a hybrid

cryptographic approach that combines ECIES and CP-ABE, ensuring both data confidentiality and fine-grained access control based on user privileges. The encrypted firmware blocks are then stored off-chain in IPFS, where each block is assigned a unique CID, enabling efficient retrieval and decentralized storage without relying on a central server.

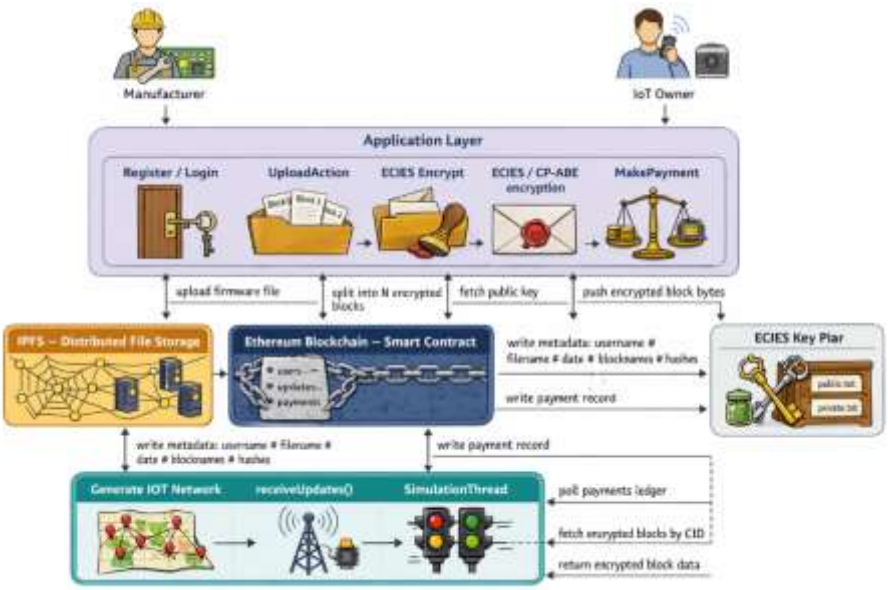


Fig. 1: Proposed system architecture

At the same time, essential metadata such as user information, firmware details, block identifiers, and corresponding hash values are recorded on the Ethereum blockchain using smart contracts, ensuring immutability, transparency, and verifiable audit trails. The IoT Owner accesses the system to browse available updates, initiate purchase requests, and complete secure payment transactions, all of which are permanently recorded on the blockchain. A dedicated key management module handles the generation, distribution, and secure storage of cryptographic keys required for encryption and decryption processes. On the IoT side, a simulation module continuously monitors blockchain transactions to verify payment status in real time. Once payment confirmation is detected, the system retrieves the encrypted firmware blocks from IPFS using their respective CIDs.

The retrieved data blocks are then decrypted using the authorized private key and reassembled to reconstruct the original firmware file, ensuring that only legitimate and authorized devices can access valid updates. Smart contracts automate critical functions such as authentication, authorization, and payment validation, eliminating manual intervention and reducing the risk of errors or fraud. Multi-threading techniques are employed within the simulation to enable concurrent processing of update requests while maintaining a responsive user interface. Additionally, visual indicators are provided to represent update progress, system status, and transaction confirmation, improving usability and monitoring capabilities. Overall, the proposed architecture delivers a secure, transparent, and decentralized update distribution mechanism, effectively addressing challenges such as unauthorized access, data tampering, and inefficient update delivery, making it suitable for applications in smart homes, healthcare IoT, and industrial automation systems.

### 3.1 ECIES

The ECIES as shown in Fig. 2 is a hybrid encryption system combining asymmetric elliptic curve cryptography with symmetric encryption to provide confidentiality, authenticity, and integrity.

Here is a detailed internal workflow of ECIES as relevant to your project:

**1. Key Generation**

- A private key $d$is randomly selected from the elliptic curve's finite field.
- The corresponding public key $Q = d \times G$is computed by multiplying the private key with the curve's base point $G$.
- The public key $Q$is distributed, while the private key $d$remains secret.

**2. Encryption Process**

- The sender wants to encrypt a plaintext message $M$for the recipient who has public key $Q$.
- The sender generates an ephemeral, random private key $k$and computes ephemeral public key $R = k \times G$.
- Using the recipient's public key $Q$, the sender computes the shared secret point $S = k \times Q$.
- From $S$, the sender derives symmetric encryption keys (such as an AES key) using a Key Derivation Function (KDF).
- The plaintext $M$is encrypted with the derived symmetric key to produce ciphertext $C$.
- A Message Authentication Code (MAC) is generated to ensure data integrity.
- The final encrypted message consists of the ephemeral public key $R$, ciphertext $C$, and MAC.

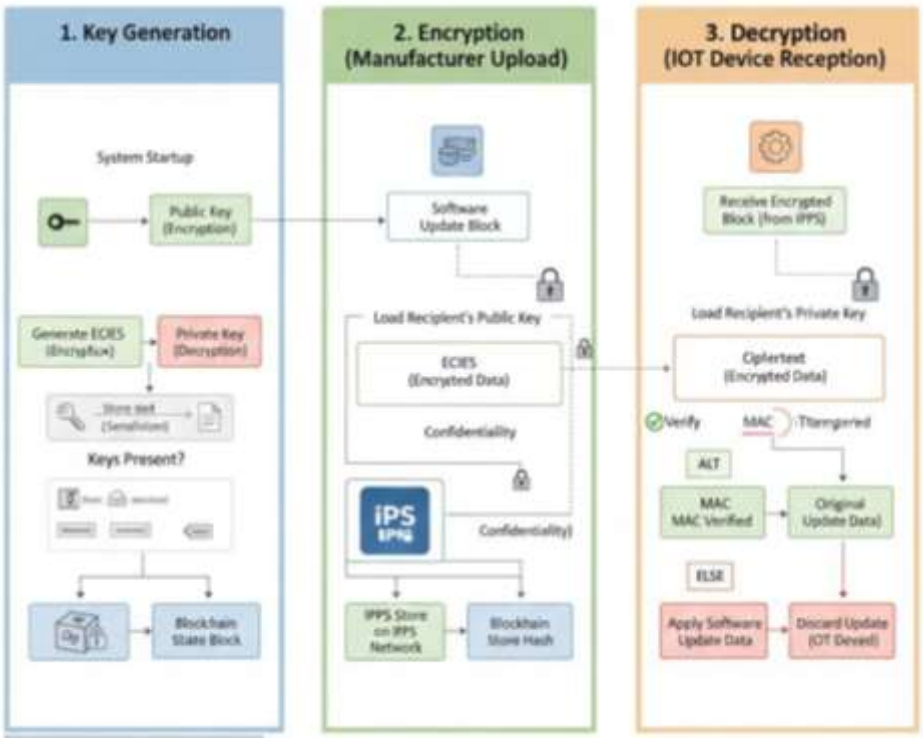


Fig. 2: ECIES asymmetric cryptography.

**3. Decryption Process**

- The recipient receives $R, C,$and MAC.
- Using their private key $d$, the recipient computes the shared secret $S = d \times R$.
- The same symmetric key is derived from $S$using the KDF.
- The ciphertext $C$is decrypted with the symmetric key to recover the plaintext $M$.
- The MAC is verified to ensure message integrity and authenticity.

**4. Security Properties**

- The use of ephemeral keys ensures forward secrecy, since each message encryption uses a new random private key.
- The combination of asymmetric key agreement with symmetric encryption provides efficiency and security.
- Inclusion of MAC ensures tamper detection.

This hybrid approach of ECIES offers strong encryption with efficiency suitable for IoT software update scenarios where resource constraints and high security are both critical. The detailed internal steps of key exchange, symmetric encryption derivation, and integrity verification form the core of its secure design.

**4. RESULTS ANALYSIS**

This research follows a structured flow that integrates IoT software update management with blockchain, decentralized storage, and cryptographic security. Below is a detailed description of the code flow:

1. **User Interaction and Web Requests (Django Views)**
    - The system starts with user registration and login via dedicated Django views for Manufacturers and Owners.
    - Upon successful authentication, Manufacturers can upload software updates, while Owners can browse available updates and make payments.
    - Each web request triggers corresponding backend operations that interact with blockchain and IPFS.
2. **File Upload and Block Processing**
    - When a Manufacturer uploads a software update file, the file is read and its size determines how it will be split into blocks.
    - The file is divided into chunks (3, 5, or 10 blocks depending on size) to optimize storage and transfer.
    - Each block is then encrypted using CP-ABE-style encryption implemented with ECIES public-key cryptography to secure update data.
    - Encrypted blocks are serialized (pickled) and uploaded to IPFS, which generates unique content hashes for each block.

2. **Blockchain Data Recording**

- Metadata such as manufacturer username, filename, upload date, block names, and IPFS hashes are composed into data strings.
- These metadata strings are saved on the Ethereum blockchain by invoking smart contract functions through web3.py.
- Similarly, payment details recorded by Owners are stored on blockchain to ensure tamper-proof transaction records.

2. **Data Retrieval and Display**

- The system supports reading user, update, and payment data by calling blockchain smart contract getter functions.
- Retrieved data (usually newline-separated and hash-separated strings) are processed and rendered in tabular HTML format for user consumption in Django templates.
- This enables Manufacturers and Owners to view their uploads, purchases, payments, and ongoing update statuses.

2. **Encryption Key Management**

- Public and private keys for encryption/decryption are generated once and stored persistently.
- The keys are loaded on subsequent transactions to encrypt new blocks or decrypt received data.
- ECIES ensures the confidentiality and selective accessibility of update data.

2. **IoT Network Simulation (Tkinter GUI)**

- A separate Python module implements the simulation of an IoT network where nodes represent IoT devices.
- Nodes are randomly generated on a canvas and their positions ensure no overlap.
- The simulation polls blockchain payment data to identify devices entitled to receive updates.
- Selected nodes flash color indicators to simulate update reception dynamically, with animations running asynchronously to maintain UI responsiveness.

The illustrated Fig. 3 depicts the screen showing purchased software updates successfully delivered to an IoT device. It describes a network view where the selected IoT device is highlighted to indicate update reception. The received software update details are displayed in a dedicated section. This interface confirms successful delivery of purchased updates. It validates the end-to-end software update process. The screen represents the final stage of the update lifecycle.

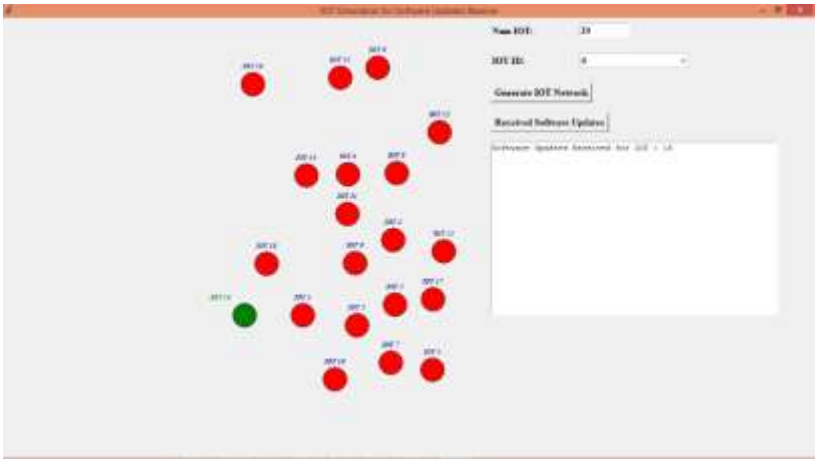

Fig. 3: purchased Software Updates

## 5. CONCLUSION

The research presents a decentralized and secure framework for IoT software updates by combining blockchain, IPFS, and advanced cryptographic methods. Ethereum-based smart contracts implemented through web3 ensure transparent, tamper-proof recording of users, updates, and transactions, improving trust and traceability. IPFS enables efficient and distributed storage of large update files, reducing dependence on centralized systems. The integration of ECIES with CP-ABE provides strong encryption with controlled access, protecting devices from unauthorized updates. A Django-based interface supports user interactions such as registration, update uploads, and payments, while a Tkinter simulation visualizes update distribution across the IoT network. The system effectively addresses issues like centralization, security vulnerabilities, and lack of transparency in traditional approaches. The modular design and tested performance demonstrate a scalable, reliable, and future-ready solution for secure IoT update management.

## REFERENCES

[1] Khan MA, Salah K (2018) Iot security: Review, blockchain solutions, and open challenges. Future Gen Comput Syst 82:395–411. https://doi.org/10.1016/j.future.2017.11.022

[2] Miller C (2011) Battery firmware hacking: Inside the innards of a smart battery. Technical report, Accuvant Labs

[3] Zetter K How the NSA's Firmware Hacking Works and Why It's So Unsettling. https://www.wired.com/2015/02/nsa-firmware-hacking/

[4] Kolias C, Kambourakis G, Stavrou A, Voas J (2017) Ddos in the iot: Mirai and other botnets. Computer 50(7):80–84. https://doi.org/10.1109/MC.2017.201

[5] Vlajic N, Zhou D (2018) Iot as a land of opportunity for ddos hackers. Computer 51(7):26–34. https://doi.org/10.1109/MC.2018.3011046

[6] Khayer, B.; Mirzaei, S.; Alavizadeh, H.; Salehi Shahraki, A. Blockchain for Secure IoT: A Review of Identity Management, Access Control, and Trust Mechanisms. *IoT* **2025**, *6*, 65. https://doi.org/10.3390/iot6040065

[7] Xu, X.; Zeng, Z.; Yang, S.; Shao, H. A Novel Blockchain Framework for Industrial IoT Edge Computing. *Sensors* **2020**, *20*, 2061. https://doi.org/10.3390/s20072061

[8] Na, D.; Kim, J.; Jeon, J.; Park, S. A Federated Blockchain Architecture for File Storage with Improved Latency and Reliability in IoT DApp Services. *Sensors* **2023**, *23*, 8569.

[9] Basudan, S. A Scalable Blockchain Framework for Secure Transactions in IoT-Based Dynamic Applications. *IEEE Open J. Commun. Soc.* **2023**, *4*, 1931–1945.

[10] Giuseppe D'Aniello, Lidia Fotia, Blockchain and AI-based methods for trust management in IoT: A comprehensive survey, Internet of Things, Volume 34,2025,101755, ISSN 2542-6605, https://doi.org/10.1016/j.iot.2025.101755.

[11] Delladetsimas, A.P.; Papangelou, S.; Iosif, E.; Giaglis, G. Integrating Blockchains with the IoT: A Review of Architectures and Marine Use Cases. *Computers* **2024**, *13*, 329. https://doi.org/10.3390/computers13120329

[12] Maftei, A.A.; Petrariu, A.I.; Popa, V.; Lavric, A. A Blockchain Framework for Scalable, High-Density IoT Networks of the Future. *Sensors* **2025**, *25*, 2886. https://doi.org/10.3390/s25092886

[13] Yohan, A.; Lo, N.W. An Over-the-Blockchain Firmware Update Framework for IoT Devices. In Proceedings of the 2018 IEEE Conference on Dependable and Secure Computing (DSC), IEEE, Kaohsiung, Taiwan, 10–13 December 2018; pp. 1–8.