



International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 2(3) (2026)



ijerst.editor@gmail.com
editor@ijerst.com

Research Paper

Design and Implementation of a Blockchain-Based Secure IoT Software Update Framework

S. Krishna Reddy¹, Mudavath Swathi¹, Gosangi Navya¹, Hozaiifa Mahmoud Fayez Abdelaziz¹, Gujja Santhosh Reddy¹

¹Department of Computer Science and Engineering, ¹Sree Dattha Institute of Engineering and Science, Nagarjuna Sagar Road, Sheriguda, Ibrahimpatnam, Rangareddy Dist, 501510, Telangana, India.

ABSTRACT

The Internet of Things (IoT) has undergone substantial development since its origin, reshaping the way devices interact, gather information, and automate operations. Although the term was officially introduced in 1999, it is rooted in earlier technological progress such as embedded systems, wireless networking, and standardized communication protocols. In the early stages, IoT devices operated with restricted computational capabilities and minimal security features, often depending on centralized infrastructures for management and software updates. Conventional update methods, including centralized servers and Over-The-Air (OTA) techniques, have presented several challenges such as single points of failure, susceptibility to cyber threats, scalability limitations, and lack of transparency. These issues threaten the reliability, security, and trust of IoT ecosystems, particularly as they expand in size and complexity. To overcome these concerns, there is a need for a resilient framework that decentralizes control, strengthens security using advanced cryptographic methods, and ensures permanent and verifiable records of update activities. The proposed architecture based on blockchain and IPFS addresses these requirements by utilizing Ethereum smart contracts to securely and transparently manage user registrations, update information, and payment records. Software updates are protected through ECIES hybrid encryption and stored on IPFS as content-addressed data blocks, ensuring integrity and eliminating reliance on centralized storage. A Django-based web application enables smooth interaction for both manufacturers and device owners, while a Tkinter-driven IoT network simulation showcases efficient update distribution guided by blockchain data. This decentralized solution offers significant improvements over traditional approaches by reducing dependence on central systems, enhancing transparency, and ensuring secure handling of update data through cryptographic safeguards. It is well aligned with modern trends that prioritize distributed trust and scalability in IoT systems. Therefore, this framework represents a notable advancement in managing IoT software updates, promoting reliability, security, and efficiency in highly connected environments.

Keywords: Internet of Things (IoT), Blockchain, IPFS (InterPlanetary File System), Smart Contracts, Software Updates.

1. INTRODUCTION

The IoT devices have become increasingly popular due to the convenience, simplicity, and wide range of advantages they provide across various aspects of daily life. This rapid growth in adoption, particularly over the past five years, has positioned IoT as a key standard for low-power and lossy networks that operate with limited resources [1]. However, this widespread usage has also made IoT devices highly attractive targets for cyber attackers [2].

According to Miessler and Smith [3], IoT systems suffer from several critical vulnerabilities, many of which are linked to poor device management. These weaknesses include insecure network services and interfaces within the ecosystem, lack of reliable update mechanisms, reliance on outdated or vulnerable

components, inadequate privacy protection, and unsafe methods of data transmission and storage. Additionally, ineffective device management practices and the use of insecure default configurations further expose these systems to risks. Improper handling and maintenance of IoT devices, whether by manufacturers or end users, can lead to a range of negative consequences affecting the devices and their overall security.



Fig. 1: A secure and cost efficient blockchain facilitated IOT software update framework

In order to protect those IoT devices from online adversaries and malware attacks, it is necessary to keep the device's firmware up to date. Firmware is a set of instruction code and commands installed in an IoT device to perform a specific task [4]. The instruction code in the firmware can be defined as set of programmed routines to handle different components and tasks in the associated IoT device. Traditionally, the new version of firmware is delivered from the device manufacturer (vendor) to each corresponding IoT device through the Internet. This traditional firmware update process is known as Firmware Over-the-Air (FOTA).

The FOTA model is based on centralized client-server architecture, in which the device manufacturer's firmware management center (FMC) acts as the server and the IoT device acts as the client during the firmware update process. However, due to the number of deployed IoT device escalates tremendously, the FMC might not be able to deliver the newest version of firmware to each deployed IoT device in a short period of time. In addition, the FMC could suffer from excessive network traffic when a lot of IoT devices simultaneously download the firmware. Moreover, each IoT device is required to always have Internet connection in order to receive the new version of firmware from the corresponding FMC. Therefore, a secure and trusted firmware update framework for IoT environment must avoid the centralized client-server architecture and should be able to distribute the new version of firmware to IoT device with limited Internet connection.

2. LITERATURE SURVEY

Behnam Khayer et al. [4] Proposed blockchain-based solutions that provided robust authentication through mechanisms such as Physical Unclonable Functions (PUFs), enhanced transparency via smart contract-enabled reputation systems, and significantly mitigate vulnerabilities, including single points of failure and Sybil attacks. Smart contracts enabled secure interactions by automating resource allocation, access control, and verification. Cryptographic tools, including zero-knowledge proofs (ZKPs), proxy re-encryption, and Merkle trees, further improve data privacy and device integrity. Despite these advantages, challenges persist in areas such as scalability, regulatory and compliance issues, privacy and security concerns, resource constraints, and interoperability. Elizabeth Nathania Witanto et al. [5] proposed a blockchain-based OCF firmware update for IoT devices. Specifically, they introduced two types of firmware update protocol, direct and peer-to-peer updates, integrated into OCF firmware update specifications. In the direct scenario, the device, through the IoT gateway, can download the new firmware update from the manufacturer's server. Meanwhile, in the peer-to-peer scheme, the device can query the update from the nearby gateways.

Alexandru A. Maftei et al. [6] introduced efficient methods for managing, transmitting, and securing data packets within a blockchain-enabled IoT network. The proposed framework uses a gateway node

to aggregate multiple data packets into single transactions, increasing throughput, optimizing network bandwidth, reducing latency, simplifying data retrieval, and improving scalability.

Almarri, S. et al. [7] examined how blockchain's decentralized, immutable, and transparent features contribute to enhancing security and trust in IoT networks. Key findings indicate that blockchain integration can prevent data manipulation, ensure robust identity management, and facilitate transparent, verifiable transactions, supporting both security and trust in IoT systems. Andreas Polyvios Delladetsimas et al. [8] reviewed lightweight consensus mechanisms to reduce computational demands. They also utilized established platforms, such as Ethereum and Hyperledger, or custom blockchains designed to meet marine-specific requirements. Additional approaches incorporated technologies such as fog and edge layers, software-defined networking (SDN), the InterPlanetary File System (IPFS) for decentralized storage, and AI-enhanced security measures, all adapted to each application's needs. Renan Yamaguti et al. [9] proposed a approach that extended the Three-Phase Methodology (TpM) by introducing an innovative entity, the IoT Operator, which acted as a custody caretaker, contract enforcer, and mediator. By leveraging blockchain's secure and immutable ledger, the IoT Operator ensures the reliable monitoring and governance of IoT applications.

3. PROPOSED SYSTEM

The proposed workflow integrates blockchain, IPFS, cryptography, and IoT simulation to manage secure software updates for IoT devices.

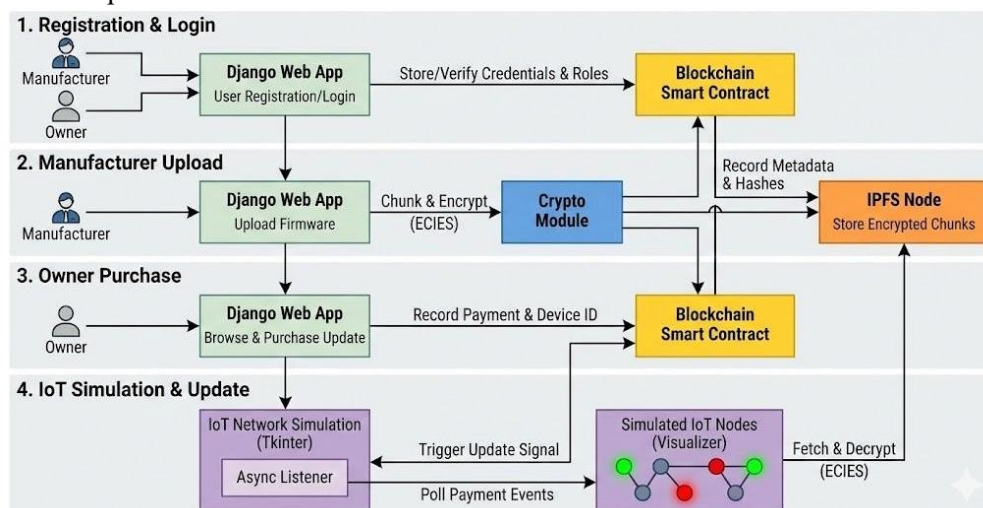


Fig. 2: Proposed system workflow

Here is the detailed workflow of proposed system:

1. User Registration and Login

- Users (Manufacturers or Owners) register through the Django web portal.
- User credentials and roles are recorded immutably on the blockchain via smart contracts.
- Users log in by validating credentials against blockchain-stored data.

2. Software Update Upload by Manufacturer

- Manufacturer logs in and uploads software updates files through the web interface.
- The update file is divided into multiple blocks depending on file size.
- Each block is encrypted using CP-ABE-like encryption with public keys.
- Encrypted blocks are serialized and stored in a decentralized IPFS network.
- IPFS returns cryptographic hashes for each block.
- Metadata including block names, hashes, manufacturer name, file name, and upload date is recorded on the blockchain for traceability and verification.

3. Viewing and Purchasing Software Updates by Owners

- Owners log in and view available update records sourced from the blockchain.
- Upon selecting a software update, owners initiate payments, which are recorded on the blockchain linked to their IoT device IDs and purchased software.
- This enables automated and transparent update transaction recording.

4. IoT Network Simulation for Update Reception

- A separate Tkinter GUI simulates the IoT network with nodes representing devices.
- Nodes are randomly placed ensuring no spatial overlap.
- The simulation polls blockchain payment records to detect updates relevant to IoT devices.
- Nodes corresponding to devices that received updates visually flash (green/red) to mimic update receipt.
- This update process animation runs on a background thread to maintain UI responsiveness.

5. Update Decryption and Use

- IoT devices/infrastructure receive IPFS links (hashes) to encrypted blocks.
- Devices use private keys to decrypt blocks using CP-ABE decryption ensuring only authorized access.
- Software update blocks are reassembled and applied to the IoT device securely.

6. Data Integrity and Security

- Blockchain acts as a tamper-proof ledger for user info, software update metadata, and payment logs.
- IPFS provides a distributed file storage system resistant to censorship and data loss.
- CP-ABE encryption ensures that only allowed recipients can decrypt and use the update data.
- The system provides transparency, security, and trustworthiness throughout the software update lifecycle.

Blockchain

Blockchain acts as the backbone for trust and transparency in managing user records, software update metadata, and payment transactions as shown in Fig. 4.3. It ensures the immutability and auditable tracking of all critical operations in the update lifecycle.

Flow:

- Establish connection to an Ethereum node using Web3.py.
- Load smart contract ABI and deployed address to instantiate contract objects.
- Upon user registration, encrypt and bundle user credentials and metadata, then invoke smart contract functions to store them immutably on-chain.
- When software updates are uploaded, record metadata such as manufacturer, filename, block names, IPFS hashes, and upload timestamps in smart contracts.
- Payment transactions triggered by owners are also recorded on-chain, linking owner identities, purchased software, IoT device IDs, and payment details.
- Blockchain retrieval functions enable querying users, updates, and payments for UI rendering and backend logic.

4. Results Description

The illustrated fig. 3 represents the file upload screen used for software update deployment. It describes an interface that enables manufacturers to select files from the local system for uploading. The page includes a file selection option and a submission control to initiate the upload process. This screen supports the transfer of encrypted software update blocks into the system. The layout ensures simplicity and controlled interaction. It plays a key role in secure software distribution.

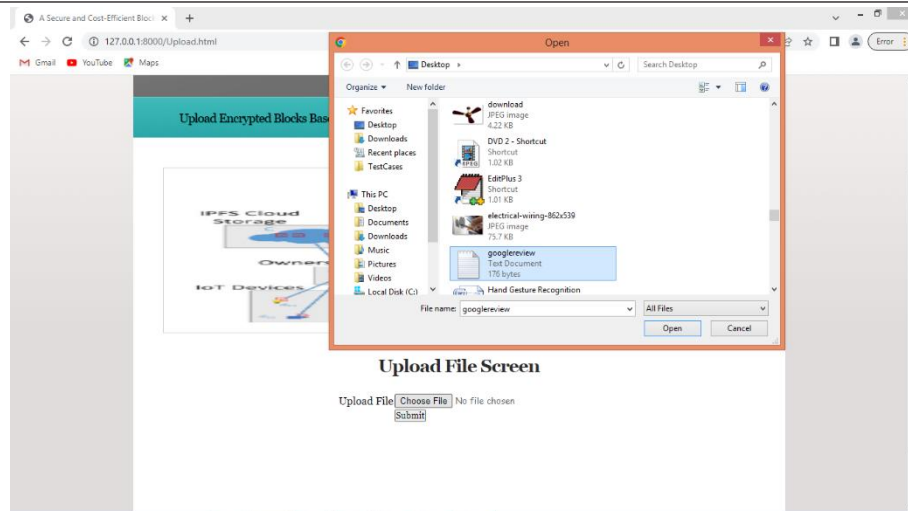


Fig 3: Uploading File

The illustrated Fig. 4 depicts the screen that displays uploaded software update files along with their generated software block details. It shows a tabular view containing manufacturer name, software filename, upload date, individual software block names, and their corresponding verification hash values. This interface enables transparent verification of software integrity after encryption and block generation. The structured presentation supports traceability of each uploaded update. It plays a key role in ensuring data authenticity and tamper resistance. This screen facilitates secure monitoring of blockchain-based software updates.

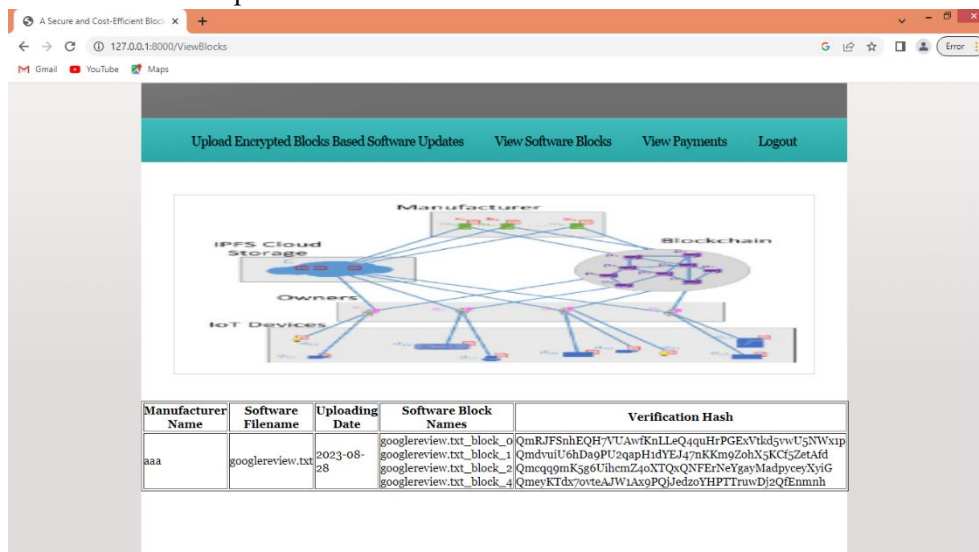


Fig. 4: Uploaded File with Software Block Names and its Hash values Screen

The illustrated Fig. 5 depicts the purchasing software updates screen for IoT owners. It shows a tabular interface listing available software updates uploaded by manufacturers. Details such as manufacturer name, software filename, and upload date are displayed for reference. A selectable purchase option allows owners to initiate the buying process. This screen supports informed decision-making before software acquisition. It enables controlled access to authorized software updates.

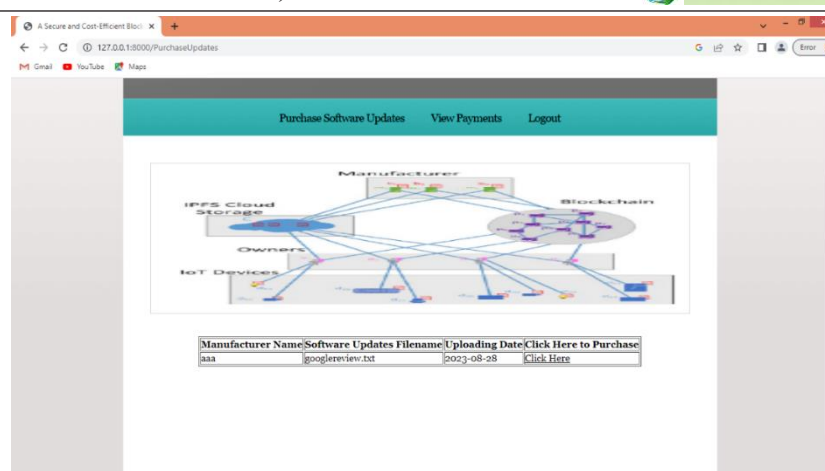


Fig 5: Purchasing Software Updates Screen

The illustrated Fig. 6 depicts the screen used to view the payment status after completing a software update transaction. It describes a confirmation interface that displays the successful completion of the payment for a selected IoT device. The screen provides feedback indicating that the transaction has been processed correctly. Navigation links allow the user to proceed with further actions or logout. This page ensures transparency in payment confirmation. It marks the completion of the purchase workflow.

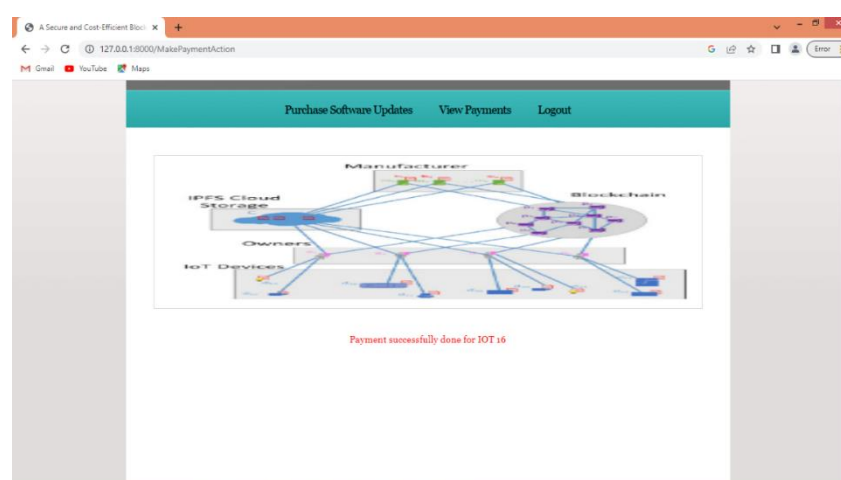


Fig 6: Viewing payment status

5. Conclusion

The research effectively presents a reliable, decentralized, and scalable approach for managing IoT software updates by combining blockchain, IPFS, and advanced cryptographic methods. Through the integration of Ethereum-based smart contracts using web3.py, the system establishes a transparent and tamper-proof ledger for managing user data, software updates, and payment activities, thereby improving trust and traceability. Utilizing IPFS as a decentralized storage solution allows large update files to be distributed efficiently without burdening the blockchain, maintaining the advantages of decentralization. The security layer, implemented with ECIES-based encryption, protects update data through controlled access, preventing unauthorized or harmful software from reaching IoT devices. The Django-based web platform supports essential functionalities such as user registration, authentication, update uploads, and payment processing. Additionally, a Tkinter-based simulation provides a visual representation of how updates are propagated across an IoT network, guided by blockchain records. Collectively, these elements resolve common issues in traditional IoT update systems, including dependence on centralized infrastructures, limited transparency, and inadequate security measures. Comprehensive testing in simulated settings confirms that the system delivers encrypted updates in a

secure and efficient manner, with both encryption processes and blockchain operations optimized for devices with limited resources. The modular structure and well-defined component separation simplify system maintenance and support future enhancements. Overall, this framework strengthens confidence among stakeholders by ensuring the authenticity of updates, reliability of transactions, and smooth delivery processes.

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] Miessler D, Smith C OWASP Internet of Things Project - OWASP. https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Vulnerabilities Accessed 10 Apr 2018
- [4] 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>
- [5] Witanto, E.N.; Oktian, Y.E.; Lee, S.-G.; Lee, J.-H. A Blockchain-Based OCF Firmware Update for IoT Devices. *Appl. Sci.* **2020**, *10*, 6744. <https://doi.org/10.3390/app1019674>
- [6] Maftai, 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>
- [7] Almarri, S.; Aljughaiman, A. Blockchain Technology for IoT Security and Trust: A Comprehensive SLR. *Sustainability* **2024**, *16*, 10177. <https://doi.org/10.3390/su162310177>
- [8] 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>
- [9] Yamaguti, R.; Ferreira, L.C.B.C.; Lui Motta, L.; Montali Assumpção, R.; Branquinho, O.C.; Iervolino, G.; Cardieri, P. IoT and Blockchain for Support for Smart Contracts Through TpM. *Sensors* **2025**, *25*, 5001. <https://doi.org/10.3390/s25165001>