Distributed blockchain-based firmware update architecture for IoT environments

Jesús Rugarcía[1], Santiago Figueroa-Lorenzo[2,3], Saioa Arrizabalaga[2,3], and Nasibeh Mohammadzadeh[2]

[1]University of the Basque Country UPV/EHU, Donostia / San Sebastián-20018, Spain
[2]CEIT-Basque Research and Technology Alliance (BRTA), Donostia / San Sebastián-20018, Spain
[3]School of Engineering, University of Navarra, Tecnun, Donostia / San Sebastián-20018, Spain

Abstract. The Internet of Things (IoT) is one of the most rapidly expanding fields of technology. IoT devices often have limited capabilities when it comes to security, and have been shown to have vulnerabilities that are often exploited by malicious agents. To fix those vulnerabilities, firmware updates are often needed. The process, however, can also be vulnerable. A secure update mechanism is needed to create a more secure IoT environment. This paper proposes a secure distributed IOT firmware update solution using Hyperledger Fabric Blockchain and IPFS based on the RFC 9019 and previously proposed frameworks, contributing with a strong manifest format and defining authentication and verification procedures. More importantly, we provide a public implementation on which performance tests were made, demonstrating the promising feasibility of using distributed ledger technologies for this problem.

Keywords: IoT, Hyperledger Fabric Blockchain, Security, Distributed solution, Firmware update.

1 Introduction

The Internet of Things (IoT) is one of the most rapidly expanding sectors of technology. The market size for IoT grew from 165 billion dollars in 2021 to 201 billion dollars in 2022. Additionally, it's expected to experience a significant surge [1], and the global market for IoT is expected to grow to about 1.39 trillion dollars by 2024 [2].

Despite advancements in IoT security, including the integration of STM32 microcontroller families [3] known for their robust security features such as secure firmware installation due to embedded secure root services, challenges persist[4]. These IoT devices often struggle with implementing strong security protocols, leaving them vulnerable to cyber threats. The frequency of cyber attacks targeting IoT has been increasing substantially. The number of attacks has been rapidly increasing year by year and has nearly doubled in 2022.

David C. Wyld et al. (Eds): NLAI, IBCOM, MLDS, GridCom, SCAI, ICCSEA, SPPR – 2024
pp. 101-125, 2024. - CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.142207

In recent years, there have been some important security threats concerning IoT devices. One of the most well-known ones was the Mirai Botnet [5], where thousands of devices were used to perform DDoS attacks. When a vulnerability is found, it is necessary to update a device's firmware in order to patch it. This process, however, is also vulnerable. Attackers can use the process to obtain firmware images and perform reverse engineering or to install images of their own [6]. A famous case of firmware update exploitation is the "Jeep Hack," where IBM researchers were able to exploit the firmware update mechanism of a microcontroller used in a Jeep car and obtained access to the vehicle's control system [7].

The increasing use of IoT devices in daily life has made them essential for critical functions across many sectors, including healthcare, transportation, and home automation. As we rely more on these devices, it's crucial to ensure their proper functioning through regular and secure firmware updates. Unfortunately, the traditional methods used for these updates often rely on centralized systems, which introduce several risks like single points of failure and are prone to sophisticated cyber-attacks. These vulnerabilities can cause significant disruptions across extensive networks of connected devices. As IoT technology continues to advance rapidly, it often outpaces the corresponding security measures, leaving devices exposed to current threats and poorly equipped to handle new challenges. Recent major security breaches highlight the urgent need to reconsider traditional security models and shift towards more robust and scalable solutions. A decentralized approach, which utilizes the inherent security features of blockchain technology, offers a compelling alternative. This method can significantly improve the reliability and security of firmware distribution processes. This paper proposes a new framework based on distributed blockchain technology, specifically designed to tackle the significant security challenges found in conventional firmware update mechanisms for IoT systems. By integrating the Hyperledger Fabric Blockchain and the InterPlanetary File System (IPFS), we aim to create a scalable and secure environment. This framework not only reduces the risks associated with central points of failure but, more importantly, ensures the integrity and authenticity of firmware updates across a wide range of IoT devices. Our approach greatly enhances the security protocols for IoT firmware updates, offering a solid safeguard against potential vulnerabilities and ensuring continued efficiency and safety in device operations.

The firmware update process presents other problems beyond the direct exploitation of vulnerabilities. For example, there are cases where the firmware author disappears from the market, leaving devices that did not download the latest updates permanently unable to obtain them [8].

Therefore, there is a need for secure and reliable firmware update methods. The Software Updates for Internet of Things (SUIT) working group of the IETF proposed a standard for IoT firmware updates in the RFC 9019 [9], which also proposes a manifest format for firmware update based on RFC 9124 ([10]) and

enables Hardware Security Modules (HSM) integration contained in RFC 4108 [11]. This proposal, however, is based on a centralized architecture, which could have some of the following problems. Centralized architectures offer a single point of failure, making them more vulnerable to DDoS attacks. They could also be vulnerable to overloads since all the dependent devices must access the updates through a single server. In the case of data loss or vendor disappearance from the market, all the devices that are dependent on the server could permanently lose access to the updates.

There are previous proposals that try to mitigate those problems using decentralized architectures based on blockchain which are described in section 2. However, these proposals are centered on the design of a framework, and leave certain proceedings to be decided in the implementation, while not offering the proposal of one. For example, they do not specify the details of the update verification or author authentication processes. Our proposal introduces the following contributions on top of previous work:

- Publicly Accessible Implementation: Our framework introduces a publicly accessible blockchain-based architecture for IoT firmware updates. By making our implementation openly available, we invite surveys, testing, and contributions from the broader research community. This openness not only enhances transparency but also encourages collaborative improvements and validation of our security protocols across diverse IoT platforms and scenarios, thereby enhancing the robustness of our solution.
- Testing and feasibility evaluation using the proposed implementation: We conducted comprehensive testing and feasibility evaluations of our proposed implementation to ensure its effectiveness and practicality in real-world scenarios. These evaluations included simulated environments where various IoT network conditions and attack scenarios were recreated to assess the resilience and performance of our framework precisely. The results demonstrate that our blockchain-based solution efficiently handles firmware updates, maintaining high security and integrity of the data, which is crucial for dependable IoT operations.
- On-chain verification of both payloads and manifests: An essential feature of our architecture is the on-chain verification of both payloads and manifests, leveraging blockchain's decentralization and immutability. This verification process ensures that all firmware updates are authenticated and remain unaltered before deployment, thereby providing a robust safeguard against common cybersecurity threats, such as spoofing and man-in-the-middle attacks.
- Proposal of JWT as an authentication method for authors: The adoption of JSON Web Tokens (JWT) for author authentication in our system offers a secure, efficient means to verify the identities of entities involved in the firmware update process. The use of JWTs, known for their compactness and ease of

transmission, makes them particularly suitable for constrained IoT environments, ensuring that updates are executed only by authenticated and authorized sources.
– Proposal of a manifest format inspired by the RFC 9124 [10]: Inspired by RFC 9124, our proposed manifest format is designed to enhance the security measures in the firmware distribution process. It specifies the firmware metadata in detail, facilitating the verification process and ensuring compatibility with the receiving IoT devices. This structured and standardized format simplifies parsing and validation, significantly reducing the risks associated with deploying corrupt or incompatible firmware.

2 Background

The background of our study intertwines the foundational principles set forth by Request for Comments (RFCs) with the innovative architectures of the InterPlanetary File System (IPFS) and Hyperledger Fabric blockchain, illustrating a convergence of standardization, decentralized storage, and secure, permissioned blockchain frameworks.

2.1 RFC

This section provides an overview of the concept of Request for Comments (RFCs) and details the specific RFCs referenced in our proposal:

RFC 9019 RFC 9019[9], titled "A Firmware Update Architecture for Internet of Things," provides guidelines and best practices for securely updating software on IoT devices. It addresses the challenges and requirements for a robust update mechanism, emphasizing the importance of security and reliability in the IoT context.

RFC 9124 RFC 9124[10], titled "A Manifest Information Model for Firmware Updates in Internet of Things (IoT) Devices," specifies the information elements required in a manifest to secure firmware updates of IoT devices. It defines the syntax and semantics of the manifest, as well as the processing rules for the target devices. It also explains how the manifest can be used with different payload formats and firmware update scenarios.

RFC 7228 RFC 7228[12], titled "Terminology for Constrained-Node Networks," defines key terms and concepts relevant to networks where nodes have limited processing power, memory, and energy resources, such as Internet of Things (IoT) devices. This standardized terminology facilitates communication and understanding among developers working on technologies and protocols specifically designed for these constrained environments.

RFC 4108 RFC 4108[11], titled "Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages," describes a method for securing firmware updates through digital signatures and encryption. This approach helps ensure the updates' integrity and authenticity, verifying that they haven't been tampered with during transmission. While not the only method, it offers a valuable tool for securing firmware updates in various scenarios.

These RFCs collectively offer a thorough guide for managing IoT devices, ensuring their security, and updating their firmware. By adhering to the standards and best practices outlined in these documents, developers and researchers can enhance the security, reliability, and functionality of IoT systems.

2.2 InterPlanetary File System (IPFS)

InterPlanetary File System (IPFS)[13] - The limited storage capacity of blockchain technology at the time of this project's development poses a significant challenge for storing complete updates on the network. While it's possible, the considerable length of the images could substantially slow down the network's transaction and consensus speed. Therefore, an alternative storage technology is used for storing the images. To avoid the inclusion of a single point of failure, using centralized storage for the images contradicts the decentralized design philosophy of the blockchain. Hence, a distributed P2P storage technology is considered appropriate. IPFS is one of the most prominent distributed storage technologies. It is a P2P system that uses content addressing to search among network nodes instead of a set address. When content is added to the IPFS network, it is assigned a Content IDentifier (CID). Nodes search for a file through its CID, and upon acquiring the content, the node stores a copy to share with its peers. The stored contents are immutable, which adds a layer of security, as images cannot be altered. Each new version of a file gets its own CID. There are two types of nodes: bootstrap nodes, which guide other connecting nodes to the network, and client nodes, which exchange information. A private network can be created by configuring bootstrap nodes. This requires generating a swarm key and adding it to all participating nodes' configurations. Nodes in a private network only interact with other nodes in the same network.

2.3 Hyperledger Fabric Blockchain (HFB)

Hyperledger Fabric [14] is a permissioned blockchain platform that offers a robust framework for developing applications or solutions for enterprises, particularly in the context of the Internet of Things (IoT). It provides a modular architecture that ensures flexibility, scalability, and confidentiality, enabling organizations to create private, permissioned networks. In these networks, participants are authenticated before joining, enhancing the overall security and privacy of the IoT ecosystem.

Key aspects of securing IoT using Hyperledger Fabric Blockchain are outlined in our proposal, including:

- **Security and Privacy**: Hyperledger Fabric enhances the security and privacy of IoT networks by providing a tamper-proof and transparent platform for recording data from IoT devices. This feature ensures data traceability and safeguards against unauthorized access or manipulation.
- **Authentication and Authorization**: It implements a lightweight mutual authentication and authorization model, ensuring that only authenticated devices can access the blockchain network. This protects sensitive data and fosters trust among interconnected devices.
- **On-Chain Process**: Refers to the practice of storing data directly on the blockchain ledger, including checksums, data pointers, and ownership information. This method secures the data against tampering and provides a reliable truth source for IoT applications.

3 Proposed Solution

The architecture proposed in this manuscript incorporates elements from RFC 9019 and builds upon previous decentralized solutions, positioning itself within the broader context of blockchain-based frameworks for IoT security. Specifically, the Hyperledger Fabric Blockchain (HFB) is utilized for its blockchain technology, chosen for its ability to facilitate the creation of fast, private, and permissioned networks. This choice is predicated on HFB's efficiency and scalability, which prevent the need for computationally intensive consensus algorithms like Proof of Work, thus improving upon HFB and presenting a compelling alternative to Ethereum for our use case. For storing firmware images, which are typically larger data files, the InterPlanetary File System (IPFS) is employed, leveraging its distributed nature to manage and disseminate these images across the network efficiently.

The solution divides firmware updates into manifests, which contain the firmware and update metadata, as well as the firmware image. Updates also contain two signatures made with the author's private key, one for the digest of the manifest and one for the digest of the payload, and the author's public key. In our architecture, on-chain processes ensure transparency and immutability. These processes, directly recorded on the blockchain, are essential for secure transactions and data veracity, essential in IoT environments where security cannot be compromised [15]. Off-chain mechanisms, on the other hand, enhance scalability and efficiency by managing large data, such as firmware images, outside the blockchain [16]. Utilizing IPFS for off-chain storage optimizes data handling without burdening the blockchain, allowing for quicker access and distribution of firmware updates. This bifurcated approach ensures our system remains robust yet agile, a necessity for the evolving demands of IoT security.

Additionally, our approach minimizes unnecessary storage operations, enhancing efficiency. While this increases the solution's workload, the Hyperledger Fabric Blockchain's (HFB) private and permissioned nature ensures that operations

remain both rapid and cost-effective, unlike public platforms such as Ethereum. This trade-off significantly boosts security without substantially impacting computational time. Verification on the blockchain, rather than off-chain, ensures the authenticity and verifiability of firmware updates prior to their delivery. Subsequently, the update, payload excluded, is securely stored, and a notification is dispatched to the register agent, maintaining the system's integrity, traceability and responsiveness.

Since firmware images can be heavy, storing them directly in the ledger can lead to longer processing times as the size of the ledger increases. For this reason, the payloads are instead stored into a private IPFS network, which provides a fast and tamper-proof distributed storage system.

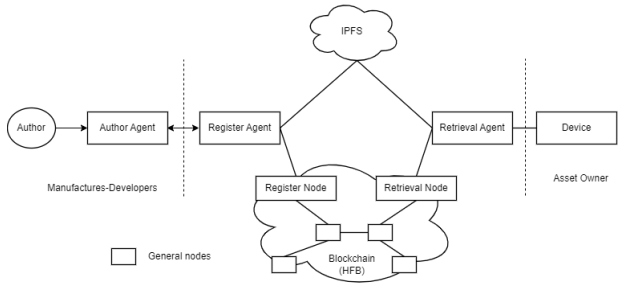

Fig. 1. Overview of the proposed architecture.

3.1 Architecture

The architecture presented in Figure 1 delineates the interaction flow between various entities involved in the firmware update process. The 'Authors', typically manufacturers or developers, originate the updates. Through the logical connections represented by dotted lines (Figure 1), they interact with 'Author Agents', which act as secure intermediaries, ensuring the updates are pushed to the 'Register Agent'. The 'Register Agent' then validates and stores these updates within the blockchain and the IPFS.

On the receiving end, 'Devices'—which are managed or owned by 'Asset Owners'—initiate the update retrieval process. This is done through 'Retrieval Agents' that fetch the latest verified updates from the blockchain and IPFS, signifying a logical link, as indicated by the dotted lines (Figure 1), which represents the request-driven communication between the devices and the system. Our architectural system includes below participants:

– **Author**: Developers or manufacturers, referred to as authors, are pivotal as the creators and distributors of firmware updates. They leverage the platform

to upload these updates, ensuring secure distribution in alignment with the guidelines established in RFC 9019 [9]. Devices across the network consume these updates, embodying the solution's end-users.

- **Author agents**: Author agents connect with a key storage, also known as keystore, that contains the author's public key and the associated registration key. They receive updates from the author and send them to the register agent.
- **Register agents**: Register agents communicate with the blockchain through chaincode and with IPFS. They receive updates from the author agent and store them after they are verified in the chaincode.
- **Retrieval agents**: Retrieval agent receives requests from devices and retrieves the corresponding last available update from the blockchain and IPFS.
- **Asset Owner**: This denotes the individual or organization that has ownership and responsibility over the 'Device', ensuring that the firmware updates are securely and efficiently managed to maintain device integrity.

The blockchain network nodes are divided into three categories:

- Register nodes attend to register petitions, both for authors and updates.
- General nodes perform the maintenance tasks of the network or store a copy of the ledger, but do not execute chaincode.
- Retrieval nodes attend to retrieval petitions and share updates with devices.

The difference between register and retrieval nodes is conceptual. Both execute the same chaincode, which has the contracts needed for registration and retrieval processes. The agents that call for the execution of the contracts are the truly limited ones, but since only they can make calls to the chaincode, the distinction between register and retrieval nodes is put into practice. The firmware update process can be divided into two key phases - phase 1, the registration phase, where the update is submitted and stored, and phase 2, the retrieval phase where IoT devices obtain and validate the update.

In Phase 1, represented in Figure 2, of the firmware update process, the author/author agent begins by creating a firmware image and manifest. They then initiate the process by requesting to register the update, providing their key, the firmware image, and the manifest to the Register Node. The Register Node takes several steps to ensure the integrity and authorization of the submission: it first verifies the author's identity to confirm they are registered and authorized to submit. Following this, it verifies the integrity of both the manifest and the firmware image. Once these steps are completed successfully, a response or error message is issued. The final step involves securely storing the manifest in the ledger (on-chain) and the firmware image on the InterPlanetary File System (IPFS) network (off-chain), ensuring the update process is secure and efficient from start to finish.

In Phase 2, represented in Figure 3, the firmware update retrieval process begins with IoT devices checking for the latest firmware version by communicating their

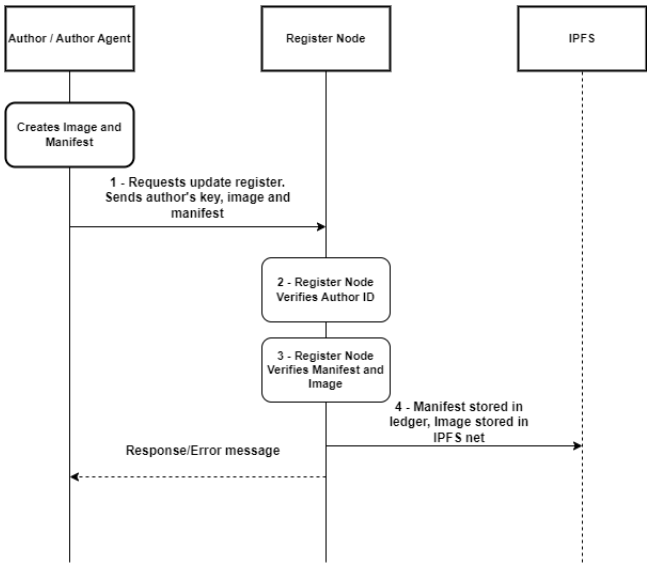

Fig. 2. Phase1: Firmware Update Registration.

details to the Retrieval Node. Upon receiving the latest version number from the Retrieval Node, if the device's firmware is outdated, it requests a new update by sending its key. The Retrieval Node then fetches the firmware image from the IPFS network using the content identifier (CID). Following this, the Retrieval Node dispatches both the manifest and the firmware image to the IoT device. The device then verifies the integrity of the manifest to ensure the update's authenticity. Once validated, the device applies the firmware update, completing the secure retrieval process. This retrieval phase allows IoT devices to securely obtain the latest updates from the nodes and validate them through the on-chain manifest before installation. The bifurcated registration and retrieval processes promote an optimized, secure firmware update mechanism using both blockchain and IPFS.

3.2 Procedure Description

The previous section represents an overview of high-level information flows. Next, the firmware update procedure is detailed, which can be segmented into the following stages: Author Registration, Firmware Update Registration, and Firmware Update Retrieval.

Author Registration As depicted in Figure 4, author registration enables creators to establish their identity on the network as a prerequisite to registering firmware updates. The process entails an author agent first sending the author's public key along with a plain text message signed by the corresponding private key

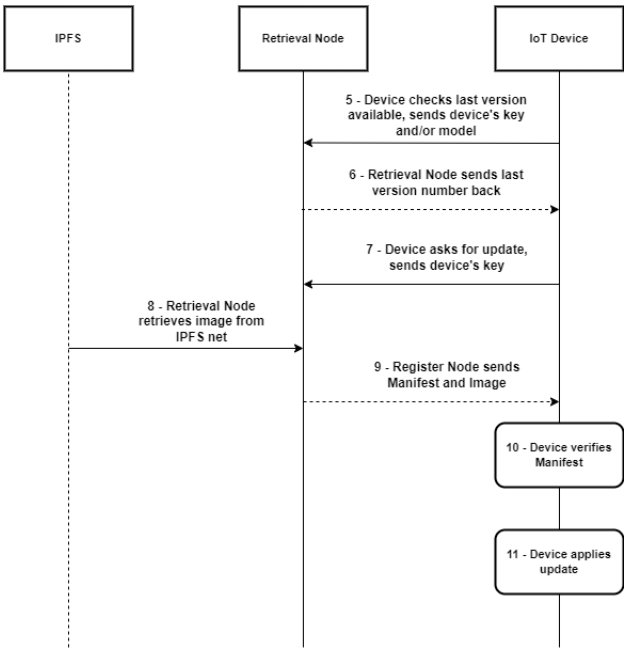

Fig. 3. Phase2: Firmware retrieval.

to the register agents as a credentials petition. This submission invokes the registration smart contract on the chaincode to verify on-chain the author's identity. If the contract can utilize the transmitted public key to cryptographically validate the signature on the message using RSA, affirming the author's ownership of the asymmetric key pair, then a temporary JSON web token (JWT) author key is generated using the same RSA infrastructure backed by on-chain secrets. This authorizes the account for subsequent update registrations. The token lifetime is 30 minutes before re-verification is required. With an authenticated key now granted and locally stored, the author can digitally sign future manifests and payloads to register verifiable firmware updates on the network, while the chaincode leverages the registered public key to validate their origin across transactions. Overall, this identity registration flow allows the system to confirm software authors before permitting update submissions using ephemeral JWTs, all enabled via public key infrastructure rooted on-chain.

Update Firmware Registration An overview of the process can be seen in Figure 5. The author begins the update registration process by creating a manifest and firmware image. The author agent then requests to register the update by sending the author's public key, the manifest, the firmware image, and the signed digests of both the manifest and the firmware image to the register agent. This request

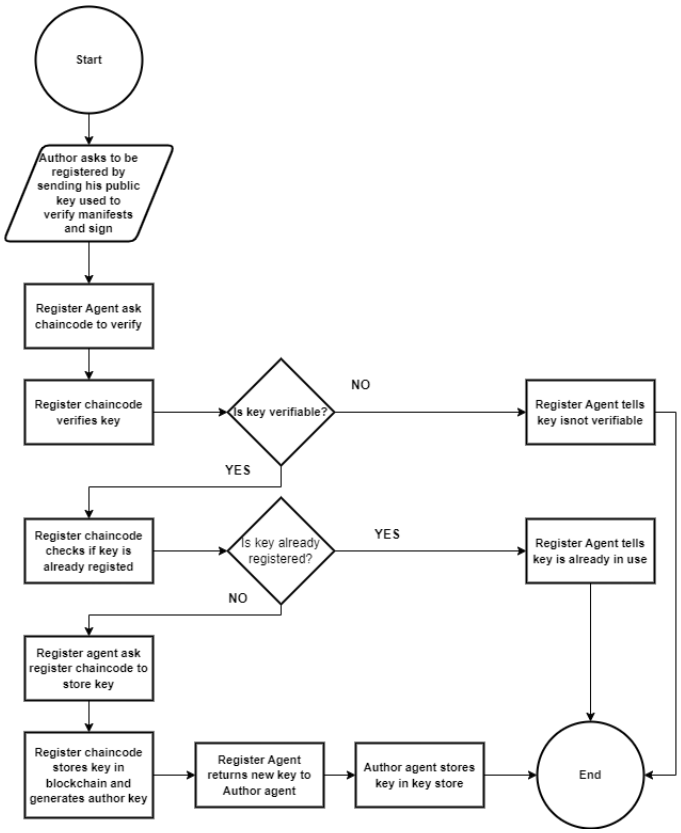

Fig. 4. Overview of the author registration flow.

includes a registration token which the register agent forwards to the chaincode. The chaincode, upon receiving the information, verifies the registration token and retrieves the author's public key. It then uses this key to verify the integrity of the manifest and the firmware image, ensuring that they have not been altered since the signing occurred. This verification of both the manifest and the firmware image together on-chain minimizes the risk of tampering and ensures that both parts are consistent with the author's original submission. After successful verification, the register agent uploads the verified firmware image to IPFS, securing a Content IDentifier (CID) for the image. This CID, along with the update details, is sent to the chaincode. The chaincode then updates its records to include the CID, effectively linking the firmware image stored on IPFS with the manifest stored on-chain. Finally, the register agent sends a success notification back to the author, confirming the completion of the update registration process.

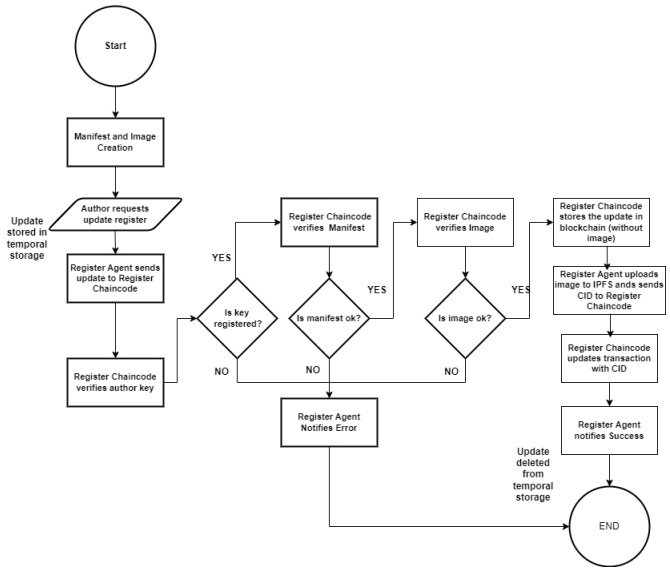

Fig. 5. Overview of the update registration flow.

Update firmware Retrieval An overview of the retrieval process can be seen in Figure 6. The device initiates the update retrieval process by transmitting its identifier, known as classID, which is provided by the manufacturer or developer (Author), along with the author's public key. This information is encapsulated in an object referred to as DeviceID, which contains the public key. Upon receiving this information, the retrieval agent forwards the request to the chaincode. The chaincode then uses the DeviceID to determine the latest updated version available for that specific device and returns this version information to the device. Upon receiving the latest version information, the device assesses whether its current firmware is outdated by comparing its version against the one retrieved. If the device determines that an update is necessary, it repeats the process of sending its DeviceID to the retrieval agent to request the full update. In response to the update request, the chaincode provides the manifest and signatures for the update, along with the Content IDentifier (CID) of the firmware image. The retrieval agent uses this CID to fetch the firmware image from IPFS, constructing an update object which is then transmitted to the device. With the update object received, the device employs the public key it stored previously, along with the received manifest and signatures, to verify the integrity of the firmware update. This verification process is crucial as it enables the device to ascertain whether the update has been altered since it was signed. Successfully verifying the update ensures that the device can detect and reject any malicious modifications, safeguarding against compromised communications between the retrieval agent and the device. Throughout this process, the retrieval agent handles three distinct types of requests: it can return the

latest version information, the manifest with signatures, or the firmware image to be updated, depending on the stage of the retrieval process and the device's requirements.

3.3 Manifest Format

Manifests play a crucial role in ensuring the integrity and security of firmware updates by providing a structured set of data that devices can rely on to authenticate and apply updates correctly. The structure we propose for manifests in this paper is heavily inspired by the framework laid out in RFC 9124[10]. This inspiration is drawn because RFC 9124 offers a comprehensive approach to secure firmware update mechanisms, which is critical for maintaining the trustworthiness and reliability of software in embedded systems and IoT devices. By adhering to a format influenced by RFC 9124, our proposed manifest structure aims to incorporate these robust security measures and best practices to enhance the update process. The specific value of aligning with RFC 9124 lies in leveraging a tried and tested framework that provides:

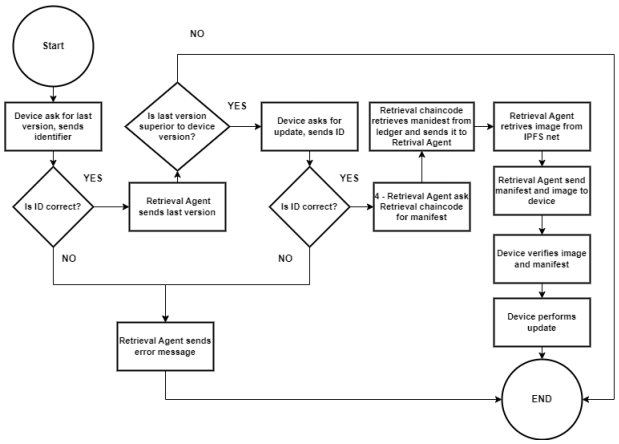

Fig. 6. Overview of the update retrieval flow.

- Enhanced Security: By detailing a structure that includes versioning, sequence numbers, and payload integrity checks, the manifest ensures that devices can verify the authenticity and integrity of an update before installation. This prevents the execution of unauthorized or malicious firmware.
- Broad Compatibility: RFC 9124's approach to manifest design considers a wide range of devices and vendors, making our proposed structure versatile across different hardware and software ecosystems.

- Future-proofing: By incorporating fields like 'VendorID', 'ClassID', and 'EncryptionWrapper', the manifest is prepared to support future developments in device identification and update encryption, ensuring long-term relevance.
- Streamlined Update Process: The detailed specification of payload format, processing instructions, and storage location simplifies the update process for devices, reducing potential errors and incompatibilities.

In summary, our manifest format's value is rooted in its comprehensive coverage of firmware update requirements, from security to compatibility, directly contributing to the goals set forth by RFC 9124. This alignment not only ensures a high standard of security and efficiency but also facilitates interoperability and future advancements in firmware update processes.

Moving from the general advantages, the specifics outlined below further detail the implementation:

- VersionID: Indicates the version of the manifest. Devices should only install updates with a superior VersionID.
- MonotonicSequenceNumber: An increasing number sequence used to verify that the manifest has been created in a time posterior to that of previous updates. We will use a UTC timestamp.
- VendorID: Optional. ID of the vendor of the device. Used to differentiate devices with identical names but different vendors. It is recommended that it uses UUID format along with ClassID.
- ClassID: Used to identify the class of devices that the update applies to. It is recommended that is uses UUID format along with VendorID.
- PayloadFormat: Describes the format of the payload.
- PayloadProcessing: Optional. Describes the steps and/or algorithms necessary to decrypt the payload.
- StorageLocation: Describes where the payload should be stored inside a given component or device.
- PayloadIndicator: Optional. Describes where the payload can be obtained from. Note that in the architecture proposed, the payload must be uploaded to the register agent alongside the manifest, making it only an option if it is available on additional storage.
- PayloadDigest: Digest of the payload used to verify its integrity when coupled with the author's key.
- Size: Size in bytes of the payload.
- AditionalInstructions: Optional. Describes additional steps to install the update.
- Dependencies: Lists all the other manifests and updates required in order to install the update.
- EncryptionWrapper: Describes the steps needed to obtain the key needed to decrypt the payload.

– Payload: Optional. Used when the payload is small enough to be shared within the manifest, like keys or configuration data.

4 Implementation Details

The design described in the previous section was implemented. An overview can be seen in Figure 7. In addition to the elements previously described, a GUI for authors was implemented. It allows interacting with the author agent more intuitively. A device-emulating app was also implemented. It allows retrieving updates connecting to the retrieval agent.

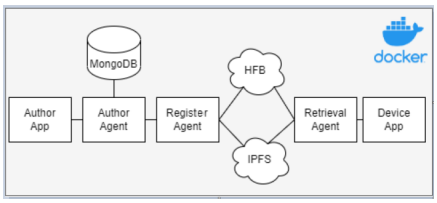

Fig. 7. Overview of the implemented architecture.

The agents and chaincode were implemented using NodeJS. The three agents serve their APIs using the Express library and use JSON and multipart/form-data for data sharing. The Axios library was used for HTTP requests. MongoDB is used to store keys in the author agent. The register and retrieval agents connect to the blockchain using the HFB gateway libraries and connect to IPFS using the JavaScript HTTP client. Docker was used to simplify the deployment of the proposed solution.

The chaincode contains three smart contracts: one for registering authors, one for update registration, and one for update retrieval. They have been developed using the HFB contract library for JavaScript.

The selected cryptographic algorithms are RSA-2048 and SHA-384. The JWT is generated using a RSA-2048 key pair stored in the chaincode containers. The 'crypto' library from NodeJS was used for digesting and signing. The 'json-stringify-deterministic' library was used to ensure that digested JSON objects had the same meta structure.

A more extensive explanation and installation guide can be found at the code repository [17].

5 Performance Evaluation

The implementation was tested to evaluate the feasibility of the solution. In this section, the testing methodology is described and the results are evaluated.

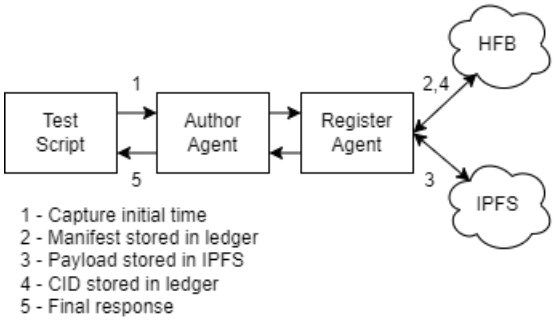

Fig. 8. Registration test structure.

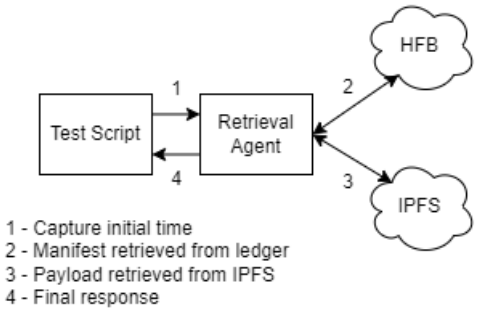

Fig. 9. Retrieval test structure.

5.1 Evaluation Methods

The register and retrieval processes are evaluated separately. A local script is made to send at least a hundred sequential petitions to the corresponding agent, containing a previously created object. For each process, four tests were run, each using a different payload. The payloads were binary files obtained from the internet. Their sizes were 552 KB, 9.5 MB, 15.4 MB, and 36.1 MB.

RFC 7228 [12] classifies IoT type 2 devices as possessing around 0.25MB of memory. The solution, however, is also intended to be used by higher-capacity devices. Therefore, tests included heavier files.

The agents logged timestamps at critical execution points, which were then analyzed through the processing of the text files.

For the registration process, in Figure 8, the logs were printed when the following steps were completed:

1. Author agent has received the petition and parsed the request, but has not processed its contents.
2. Register agent received a success message after the manifest has been uploaded to the ledger.

3. Register agent received CID from IPFS.
4. Register agent received a success message after the CID has been updated in the ledger.
5. The author agent has completed the process and is about to send the final response.

For the retrieval agent, in Figure 9, on the other hand, the logs were printed when the following steps were completed:

1. Retrieval agent has received the petition and parsed it, but has not processed its contents.
2. Retrieval agent received manifest from the ledger.
3. Retrieval agent received payload from IPFS.
4. Retrieval agent has completed the process and is about to send the final response.

The implementation was deployed on a Proxmox virtual machine which had 9.77GB of RAM and 50GB of storage available. It ran Ubuntu 20.4, used SeaBIOS, used an i440fx machine, and had 2 cores.

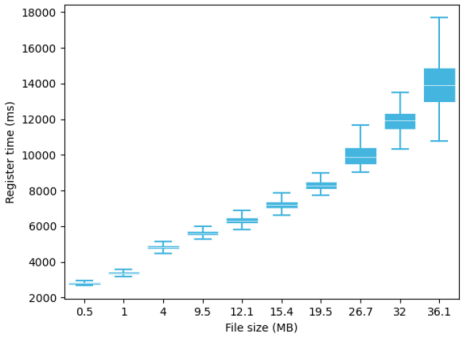

Fig. 10. Total registration time.

5.2 Evaluation Results

Figure 10 and Figure 11 show the times for the registration process. Although we can appreciate a significant increase in time as the expected payload size increases, the registered times show the feasibility of using the proposed solution and expanding on it. The on-chain verification and sharing of data between multiple parties (Agents, HFB nodes, IPFS nodes) contribute heavily to the increase in time as can be seen in Figure 11. Nevertheless, we consider that for these file sizes, the security value of on-chain verification far weights the costs.

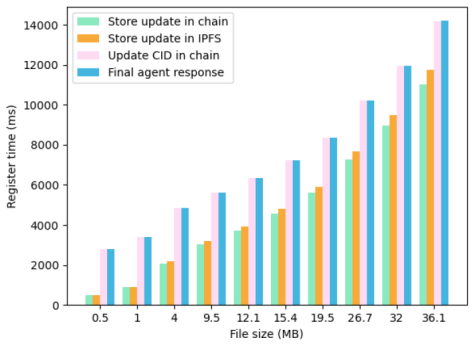

Fig. 11. Average registration time by step.

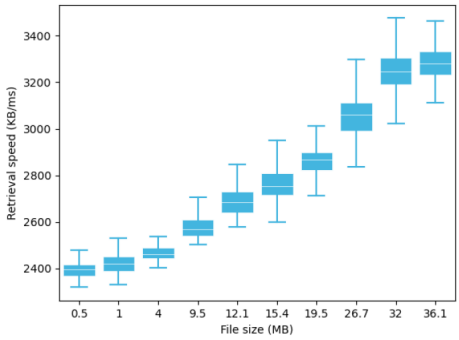

Fig. 12. Total retrieval times.

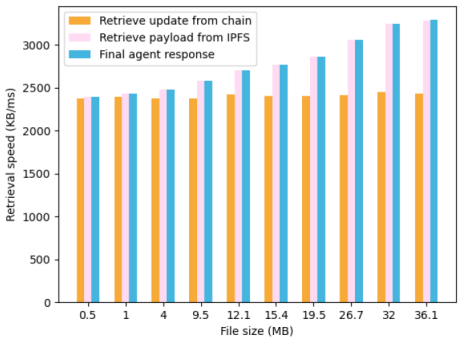

Fig. 13. Average retrieval times by step.

The retrieval process, on the other hand, remains more constant as payload size increases, as shown in Figure 12 and Figure 13. Since the retrieval process is the one that will be executed with higher frequency, it further adds to the affirmation that the solution presents itself as feasible for real-world implementations and/or

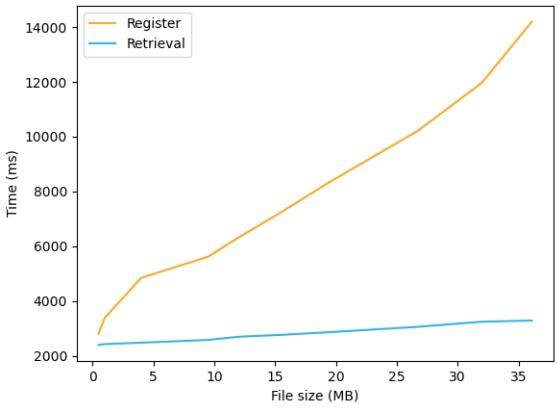

Fig. 14. Comparison between total process times.

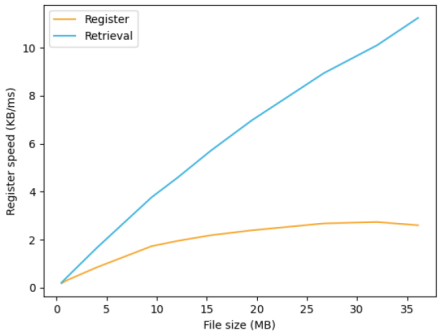

Fig. 15. Comparison between process times per MB.

further research. A comparison for both processes is shown in Figure 14. In Figure 15, we can see the time it takes to process a kilobyte in milliseconds, for both the registering and retrieval process. It is appreciable that the solution becomes more efficient as file size increases, especially for the retrieval process, which is the most benefited by it, as the IoT devices have lower capabilities than the author's computers. At the largest file size tested, 36.1 MB, we can appreciate efficiency beginning to decline for the register process, due to the limitations of the default HFB endorser node running inside the virtual machine. Future research may explore the performance impact of customized HFB network architectures and different execution environments for the solution.

6 Related Work

Previous proposals have tried to solve the problems of the IoT firmware update mechanisms. One of the most relevant ones is that of RFC 9019 [9], which proposed the use of manifests and the use of a centralized server that monitors devices.

However, this centralized approach can introduce significant challenges. Centralized systems are known to create single points of failure, making the network more vulnerable to targeted attacks that could compromise the server. This vulnerability is particularly critical in IoT systems where a single point of failure can have far-reaching impacts on security, privacy, transparency, and data integrity [18]. Scalability is another concern, as the central server may become overwhelmed with the increasing number of IoT devices requiring updates, leading to potential bottlenecks. In addition, centralized cloud servers in IoT systems can result in data exposure, posing a severe threat to data privacy and security [19]. In order to mitigate them, some proposals use distributed architectures. B.Lee et al. [20] was the first to propose the use of a decentralized architecture using a blockchain similar to Bitcoin, in which nodes are run inside IoT devices and use Proof of Work as a consensus algorithm. A. Boudguiga et al. [21] proposed the use of external nodes maintained by trusted institutions, reducing device workload. A. Yohan et al. [22] proposed an independent architecture that contains full nodes that maintain the network and light nodes that obtain updates for devices using Ethereum. M.Son et al. [23] proposed the use of Hyperledger Fabric Blockchain (HFB), enabling faster transactions. They also introduced InterPlanetary File System (IPFS) as a storage alternative to reduce block sizes. S. Choi et al. [8] proposed the separation of responsibilities between registering and retrieving nodes, and described the author's disappearing problem.

Narender et al. [24] delve into enhancing the security of firmware updates in embedded systems through blockchain technology. Their approach introduces a peer-to-peer firmware distribution network, which significantly reduces reliance on centralized servers. This decentralized model not only streamlines the distribution process but also enhances security by utilizing blockchain to verify the authenticity of firmware updates, ensuring that only legitimate and authorized updates are distributed across the network [24]. This model marks a substantial advancement in firmware update security for embedded systems, offering a more efficient and secure alternative to traditional centralized server-based systems and aligning with the evolving needs of IoT environments.

Adding to these discussions, Tsaur et al. [25] propose a blockchain-based mechanism to secure IoT firmware updates, focusing on mitigating vulnerabilities through a distributed database and improving system security and efficiency. This study emphasizes the significance of blockchain in verifying and authenticating firmware updates, ensuring devices receive genuine updates, and exploring a manifest format that enhances IoT device security.

In addition to the strategies mentioned above, the Firmware-Over-the-Blockchain (FOTB) framework represents another progressive step in the decentralized management of IoT firmware updates. FOTB uses blockchain technology to tackle some of the critical weaknesses in traditional update methods, like central points of failure, which can be risky in terms of cyber attacks and data leaks. By spreading out firmware updates over a decentralized network, FOTB increases the overall transparency and security of these systems. The setup works on a peer-to-peer basis, reducing the reliance on central servers that can become overloaded or targeted by hackers. FOTB is also versatile in how it handles updates. It supports both automatic updates (PUSH) and on-request updates (PULL), making it adaptable to different IoT environments where connection quality may vary. The framework clearly defines roles for everyone involved, from the device manufacturers to third-party service providers, ensuring that every firmware update is checked and approved through blockchain consensus. This not only strengthens security but also helps the system scale smoothly as the number of IoT devices grows, addressing some of the traditional challenges with performance and security [26].

In summary, the related works section extensively discusses various approaches to IoT firmware updates, focusing mainly on the vulnerabilities of centralized models and the benefits of decentralized solutions. Centralized systems, while common, are plagued by significant drawbacks such as single points of failure and scalability issues, which are particularly problematic in critical IoT applications. In response, recent research has shifted towards decentralized and blockchain-based architectures to mitigate these risks. These studies underscore the use of blockchain to provide a more secure and transparent mechanism for firmware updates, reducing reliance on central servers susceptible to attacks. Building on these foundational ideas, our proposal introduces a novel blockchain-based framework that not only addresses the inherent security flaws of traditional systems but also improves scalability and efficiency. By integrating the InterPlanetary File System (IPFS) with the Hyperledger Fabric Blockchain, we enhance the robustness of firmware distribution across a diverse array of IoT devices and environments. This not only streamlines the update process but also ensures the integrity and authenticity of the data exchanged, making it a formidable solution against the backdrop of escalating security demands in the IoT landscape. While many existing proposals contribute to a framework for secure updates, they often lack detailed implementation or performance metrics. Table 1 showcases how our proposal leverages these insights to address verification, authentication, and scalability within IoT firmware updates, demonstrating a comprehensive and practical approach to enhancing IoT security.

Table 1. Comparison of blockchain-based proposals for firmware update

Feature	Blockchain Technology	Verification	Authentication	Manifest Format	Implementation	Feasibility Test
B.Lee [20]	Bitcoin	On-chain	Proof Of Work	-	Public	-
A.Boudguiga [21]	Bitcoin	External Institutions	-	-	Public	-
A.Yohan [22]	Ethereum	-	-	-	Public	-
M. Son [23]	Hyperledger Fabric	-	FabricCA	-	Private	-
S.Chi [8]	-	Manifest On-chain	-	-	-	-
Narender [24]	Generic Blockchain	Smart contracts & Consensus	Device and Transaction Verification	Cryptographic hashes and signatures	Public	Real-world application and testing
Tsaur [25]	Ethereum	On-chain	-	-	Public	-
FOTB [26]	Ethereum	Smart contracts & Consensus	Smart contracts & Consensus	Metadata & Signatures	Public	Tested in simulated environment
Our Proposal	Hyperledger Fabric	Full update On-chain	JWT, RSA	Based on RFC 9124	Private	Tested on Implementation

7 Conclusions and Future Work

The data stored within the solution is guaranteed not to be tampered with as it is verified on-chain and both HFB and IPFS have strong anti-tampering mechanisms. The use of JWT after verification of keys adds a layer of security while maintaining ease of use. The use of a strong manifest format allows devices to perform more exhaustive verifications but does not significantly increase execution times, which the test has shown to be feasible for real-world implementations. While the presented solution and implementation are not flawless, this paper has shown the promising viability of a distributed architecture for IoT firmware update solutions.

Further research is needed to reach a completely secure real-world mature solution. We have identified the following areas to be among the most relevant:

- Improve CI/CD (Continuous Integration and Continuous Delivery/Continuous Deployment) practices for embedded systems by associating our approach with the delivery of firmware updates on devices in production, i.e. in the hands of asset owners and the automation of firmware delivery from HSM-based solutions such as the one presented by the STM32 family of microcontrollers.
- Integrate SBOM (Software Bill of Materials) into our approach to improving the security and integrity of firmware updates on devices through early detection of vulnerabilities and threats at both the code and infrastructure levels, i.e., throughout the software development lifecycle.

Acknowledgment

The authors would like to acknowledge the support of several projects and funding entities. This research was funded by the Centro para el Desarrollo Tecnológico y la Innovación E.P.E. (CDTI) through the project "CICERO – Contramedidas inteligentes de ciberseguridad para la red del futuro" with grant number CER-20231019.
Additional support was provided by the project "BEACON: KK-2023/00085—cyBErsecure industriAl Computing cONtinuum".

References

1. IoTAnalytics, "Global IoT market size to grow 19% in 2023 — IoT shows resilience despite economic downturn," 2023. [Online]. Available: https://iot-analytics.com/iot-market-size/
2. Statista, "Internet of things (IoT) worldwide," 2024. [Online]. Available: https://www.statista.com/outlook/tmo/internet-of-things/worldwide
3. STM32 MCU, "Security features on STM32H5 MCUs," 2024, [Online; accessed 30-Jan-2024]. [Online]. Available: https://wiki.stmicroelectronics.cn/stm32mcu/wiki/Security:Security_features_on_STM32H5_MCUs
4. SonicWall, "SonicWall CyberThreat Report," 2023. [Online]. Available: https://www.sonicwall.com/medialibrary/en/white-paper/2023-cyber-threat-report.pdf
5. M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the Mirai Botnet," in 26th USENIX Security Symposium (USENIX Security 17). Vancouver, BC: USENIX Association, Aug. 2017, pp. 1093–1110. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis
6. M. Bettayeb, Q. Nasir, and M. A. Talib, "Firmware update attacks and security for IoT devices: Survey," in Proceedings of the ArabWIC 6th Annual International Conference Research Track, ser. ArabWIC 2019. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3333165.3333169
7. C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," 2015.
8. S. Choi and J.-H. Lee, "Blockchain-based distributed firmware update architecture for IoT devices," IEEE Access, vol. 8, pp. 37 518–37 525, 2020.
9. B. Moran, H. Tschofenig, D. Brown, and M. Meriac, "A Firmware Update Architecture for Internet of Things," RFC 9019, Apr. 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9019
10. B. Moran, H. Tschofenig, and H. Birkholz, "A Manifest Information Model for Firmware Updates in Internet of Things (IoT) Devices," RFC 9124, Jan. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9124
11. R. Housley, "RFC4108: Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages," Internet Engineering Task Force, August 2005, RFC 4108. [Online]. Available: https://www.rfc-editor.org/info/rfc4108
12. C. Bormann, M. Ersue, and A. Keränen, "Terminology for Constrained-Node Networks," RFC 7228, May 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7228
13. Protocol Labs, "Interplanetary file system (IPFS)," IPFS Documentation, 2024, accessed: yyyy-mm-dd. [Online]. Available: https://ipfs.tech/

14. H. Honar Pajooh, M. Rashid, F. Alam, and S. Demidenko, "Hyperledger fabric blockchain for securing the edge internet of things," Sensors, vol. 21, no. 2, p. 359, 2021.
15. Y. Tsang, C. Lee, K. Zhang, C. Wu, and W. Ip, "On-chain and off-chain data management for blockchain-internet of things: A multi-agent deep reinforcement learning approach," Journal of Grid Computing, vol. 22, no. 1, pp. 1–22, 2024.
16. G. Wood et al., "Ethereum: A secure decentralised generalised transaction ledger," Ethereum project yellow paper, vol. 151, no. 2014, pp. 1–32, 2014.
17. J. Rugarcía, S. Figueroa-Lorenzo, and S. Arrizabalaga, "Code repository," 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7788715
18. H. F. Atlam, M. A. Azad, A. G. Alzahrani, and G. Wills, "A review of blockchain in internet of things and AI," Big Data and Cognitive Computing, vol. 4, no. 4, p. 28, 2020.
19. H. H. Pajooh, S. Demidenko, S. Aslam, and M. Harris, "Blockchain and 6G-enabled IoT," Inventions, vol. 7, no. 4, p. 109, 2022.
20. B. Lee and J.-H. Lee, "Blockchain-based secure firmware update for embedded devices in an internet of things environment," J. Supercomput., vol. 73, no. 3, p. 1152–1167, 2017. [Online]. Available: https://doi.org/10.1007/s11227-016-1870-0
21. A. Boudguiga, N. Bouzerna, L. Granboulan, A. Olivereau, F. Quesnel, A. Roger, and R. Sirdey, "Towards better availability and accountability for IoT updates by means of a blockchain," in 2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), 2017, pp. 50–58.
22. A. Yohan and N.-W. Lo, "An over-the-blockchain firmware update framework for IoT devices," in 2018 IEEE Conference on Dependable and Secure Computing (DSC), 2018, pp. 1–8.
23. M. Son and H. Kim, "Blockchain-based secure firmware management system in IoT environment," 2019 21st International Conference on Advanced Communication Technology (ICACT), pp. 142–146, 2019.
24. N. Chinthamu, M. Prasad, A. J. Chinchawade, K. K. S. Liyakat, K. Deepti, M. Karukuri, and C. M. Kumar, "Self-secure firmware model for blockchain-enabled IoT environment to embedded system," Eur. Chem. Bull, vol. 12, p. S3, 2023.
25. W.-J. Tsaur, J.-C. Chang, and C.-L. Chen, "A highly secure IoT firmware update mechanism using blockchain," Sensors, vol. 22, no. 2, p. 530, 2022.
26. A. Yohan and N.-W. Lo, "FOTB: a secure blockchain-based firmware update framework for IoT environment," International Journal of Information Security, vol. 19, no. 3, pp. 257–278, 2020.

Authors

Jesús Rugarcía earned his bachelor's degree in Software Engineering from the University of the Basque Country (UPV/EHU). He has worked on a project aimed at developing a distributed blockchain-based firmware update architecture for IoT devices, showcasing his skills in tackling complex technological challenges. Jesus is interested in Python, JavaScript, and has experience with Node.js, as well as web development frameworks like React and Express.

Santiago Figueroa Lorenzo is a researcher at CEIT, collaborating professor at TECNUN (University of Navarra) and associate director of the Data Analysis and Information Management group (DAIM). He obtained the Master in Telecommunication and Telematics Systems by the Faculty of Engineering of Havana (University of Havana) in 2012 and the PhD in Applied Engineering by TECNUN (University of Navarra) in 2021 with specialization in the line of cybersecurity in identity and access management issues in Industrial Internet of Things environments. His current research areas are Security in ICS and IoT systems, Identity and Access Management of systems, Privacy Protection and automation of the Secure Software Development Life Cycle.

Saioa Arrizabalaga received the M.Sc. degree in telecommunication engineering from the Faculty of Engineering, University of the Basque Country, Bilbao, in 2003, and the Ph.D. degree in engineering from TECNUN, in 2009. She is currently a Lecturer with TECNUN, University of Navarra, and a Researcher with CEIT. She is also the Head of the Data Analytics and Information Management Research Group. She is involved in the participation of European research projects, direction of doctoral theses, scientific, and technical publications in national and international journals.Her current research areas are data analytics and cybersecurity.

Nasibeh Mohammadzadeh works as a researcher at CEIT. She completed her master's degree with a focus on network security within cellular networks at the University of Hyderabad, India, where her project was implemented at the Institute for Development and Research in Banking Technology (IDRBT). Nasibeh pursued her Ph.D. in the Information Security Group (ISG) at Universitat Politècnica de Catalunya (UPC), focusing on blockchain technology and security, notably exploring invoice factoring through blockchain. Her passions lie in cybersecurity and blockchain technology. Currently, her research efforts are directed towards identity and access management (IAM) and Security in IOT technologies.