Security-enhanced firmware management scheme for smart home IoT devices using distributed ledger technologies

Wijesundara, W. M.A.B.; Lee, Joong Sun; Tith, Dara; Aloupogianni, Eleni; Suzuki, Hiroyuki; Obi, Takashi

Published in:
International Journal of Information Security

DOI:
10.1007/s10207-024-00827-x

Publication date:
2024

Document Version
Publisher's PDF, also known as Version of record

International Journal of Information Security (2024) 23:1927–1937
https://doi.org/10.1007/s10207-024-00827-x

REGULAR CONTRIBUTION

Security-enhanced firmware management scheme for smart home IoT devices using distributed ledger technologies

W. M. A. B. Wijesundara[1] · Joong-Sun Lee[2] · Dara Tith[3] · Eleni Aloupogianni[1] · Hiroyuki Suzuki[4] · Takashi Obi[2]

Published online: 7 March 2024

Abstract
With the increase of IoT devices generating large amounts of user-sensitive data, improper firmware harms users' security and privacy. Latest home appliances are integrated with features to assure compatibility with smart home IoT. However, applying complex security mechanisms to IoT is limited by device hardware capabilities, making them vulnerable to attacks. Such attacks have recently become frequent. To address this issue, we developed a secure verification mechanism for firmware released by the device's manufacturer. We proposed an IoT gateway for secure firmware verification and updating for smart home IoT devices utilizing the IOTA MAM (Masked Authenticated Messaging) protocol and a distributed file system with IPFS (Inter-Planetary File System) protocol. These two communication protocols ensure decentralized communication and firmware file distribution between the IoT device vendor and the IoT end device. The proposed scheme securely shares latest firmware content over IOTA and IPFS networks, performs a secure firmware update on IoT end devices and ensures authenticity and integrity of the firmware. Two types of validation methods were proposed for firmware updating and validation. We implemented the proposed scheme using three entities, Vendor, IoT gateway, and IoT end device. Our system yielded promising results in performing secure automated firmware updates on IoT end devices with very low computational power. The system's functionality was implemented using IOTA's MAM run on Raspberry Pi as an IoT gateway along with an ESP8266 Wi-Fi microcontroller, demonstrating the effectiveness of our approach. Our proposed methodology can be used for secure firmware distribution on home IoT applications.

Keywords Internet of things · Distributed ledger technologies · IOTA · Information security · Communication systems · IPFS

1 Introduction

The Internet of Things (IoT) is applied in a wide variety of fields with its ability to offer a close interconnection between things and people [1, 2]. Some of the popular applications of IoT include smart homes, wearable health devices [3, 4], smart cities, connected cars, smart sensors [4, 5], etc. With the exponential increase of connected devices and data volume, security has become an escalating challenge in IoT. Specifically, introduction of new security updates to IoT devices after the manufacturing stage is not only difficult but also often overlooked [6–8].

Cyber-attacks on IoT devices themselves and their applications are increasing by the day [9, 10]. An IoT device's performance directly depends on its firmware; thus, regularly updating the firmware is essential for the device's proper operation and safety, as well as for the introduction of new features, communication procedures, and bug fixes [11, 12]. During malicious attacks on firmware, attackers can disturb the regular functionality of the device, take remote control or violate the owner's privacy [9, 13, 14]. Although it may not be the solution to all types of cyber-attacks, appropriate firmware patching and update validation are capable of mitigating the majority of attacks that target vulnerabilities publicly known.

✉ W. M. A. B. Wijesundara
wijesundara.w.aa@m.titech.ac.jp

1 Department of Information and Communications Engineering, Tokyo Institute of Technology, Tokyo, Japan

2 Institute of Innovative Research, Tokyo Institute of Technology, Yokohama, Japan

3 Faculty of information, University of Namur, Namur, Belgium

4 Center for Mathematics and Data Science, Gunma University, Gunma, Japan

Some of the most prominent security challenges for IoT devices connected to the internet include firmware that is not up to date [15, 16], weak authentication and microchips, which are not well secured [16]. In general, IoT vendors maintain their own firmware repositories, which contain their latest firmware [17]. Whenever vendors identify security vulnerabilities, patch upgrades are distributed to the devices. However, the frequency of deploying such upgrades is often insufficient compared to the actual requirement. This can be attributed to limited profitability to manufacturers and practical difficulties [18]. Updating IoT device firmware can be carried out in two ways: through a direct communication link or remotely. However, both methods present challenges [8]. The first option requires constant physical access to the device, while the latter can be compromised by man-in-the-middle (MitM) [19] or denial-of-service (DoS) attacks [20]. In order to authenticate and validate firmware upgrades, vendors utilize digital signatures [6, 18]. Proposed schemes include encrypting the firmware image and signing using JSON (JavaScript Object Notation) and JOSE (JSON Object Signing and Encryption), to authenticate even untrusted repositories [21], or partitioning the firmware into blocks and chaining them with hash values [14]. However, most IoT devices do not support built-in mechanisms for secure firmware update and the implementation of complex security mechanisms is limited by device hardware capabilities [17, 22–24]. Therefore, timely update of authenticated firmware over-the-air remains the only viable solution for most IoT devices. The majority of current IoT solutions including firmware updating tools, adopt a centralized client–server architecture where vendors' cloud servers are connected via the internet and, all data are managed in a centralized manner by a server [1, 9, 25, 26]. As a result, when a firmware patch is made available, distributed denial-of-service (DDoS) issues are a concern [8, 27]. This in turn will largely burden internet server providers (ISPs), inter-ISP business relationships and the internet backbone [6, 18, 28]. Furthermore, centralized architecture allows single point of failure (SPOF) to occur and is vulnerable to issues such as natural disasters and local power failures [28]. Additionally, it is possible to tamper with files and update history within the centralized architecture, causing a decrease in reliability [29].

Therefore, distributed ledger technology (DLT) offers a better approach for patch delivery while simultaneously decreasing bottlenecks and liabilities associated with a centralized architecture [28, 30]. The distributed database is governed by a consensus protocol executed across the nodes of a distributed network, enabling the identification of any unintended activity [3]. Blockchain is a DLT that has been proposed for remote firmware updating [7, 8, 31–33]. However, low throughput, scalability issues, resource utilization, storage limitations, and considerable fees are a significant hindrance for the industrial and financial use cases of blockchain [3, 16, 34–36]. Moreover, a verification scheme for the firmware version is not available on the blockchain. This can be solved by polling random network nodes for present firmware versions [37]. In this case, all network nodes are expected to store all published firmware binaries, causing the block size to gradually increase with new versions over time. Alternatively, rights to download an update can be handled through a smart contract and updates can be bundled in one transaction using an aggregate signature scheme to reduce resource consumption [38]. Another study enhanced this concept with a PUSH-based method that can assist in firmware integrity preservation [39]. However, once an update is pushed to the IoT gateway, the update is forwarded from the gateway to devices without any verification, providing a window for malicious attack.

The directed acyclic graph (DAG) is another form of DLT which does not possess the limitations of blockchain [3, 16, 40]. DAG-based technology can be effectively used to build a secure, cost-effective, scalable, and faster substitute for blockchain-based distributed ledgers [41, 42]. New distributed ledgers like IOTA, Algorand, hashgraph, and Ouroboros have been introduced [3, 16, 40, 43, 44]. IOTA is a distributed ledger that is open-source, does not suffer from scalability issues and facilitates microtransactions for the IoT free of charge [45]. An IoT gateway for secure firmware verification and updating for devices can be managed utilizing the IOTA Masked Authenticated Messaging (MAM). Additionally, IOTA's DAG, or in other words, "The Tangle", is a tamper-resistant and permission-less data repository [3, 46–48]. In contrast to the high cost for storing large files on blockchain, storage of large files in DAG-based decentralized architecture is assured through the Inter-Planetary File System (IPFS) [29, 49, 50].

This study introduces a novel solution to the problem of IoT device firmware management, to handle vulnerabilities in update mechanisms, authentication mechanisms and large-scale firmware deployments. Considering security, cost and latency issues associated with previous studies, we proposed a firmware updating and authentication scheme based on IOTA MAM and IPFS. To the extent of our knowledge, this is the first work that uses IOTA MAM for firmware updating and authentication and IPFS for firmware binary distribution. A proof of concept was implemented and tested against IOTA Tangle to authenticate the integrity and authenticity of IoT device firmware, IPFS for decentralized firmware content sharing and keep it up to date using the proposed scheme. Our system was able to perform secure automated firmware updates on IoT end devices with very low computational power and latency. The methodology used in this study can be applied for secure firmware distribution on home IoT applications [6, 17, 18].

In this paper, we give a brief introduction followed by the current literature related to firmware updating and com-

ponents and framework of our proposed firmware updating method. In the end, we discuss the outcomes of our study, comparing our results with previous work.

2 Related work

Lee et al. (2017) [18] have proposed blockchain to obtain the authenticity of firmware by letting each IoT device act as a normal node connected through the blockchain network. If a normal node is requesting firmware information, it becomes a request node. A normal node also can validate its firmware and respond to request nodes. In that case, the normal node will become a response node. Verification node is hosted by the vendor for validating firmware of the normal node. While their scheme promised secure firmware verification, their system has to go through PoW, which leads to high resource utilization, latency and scalability issues. While this model has not been implemented and analyzed in practice, they also do not propose a method to verify the integrity of the firmware file [18]. Boudguiga et al. [37] proposed that every IoT device could occasionally poll random network nodes for firmware versions present, as a method to update device firmware. Firmware versions when freshly released should be first verified by peer nodes in the blockchain using consensus protocol. If any IoT device from a particular vendor intends to update its firmware, the device must generate a request for firmware update.

In Boudguiga et al. 's method [37], the firmware has first to be verified by peer nodes, for devices to be able to download firmware from the respective vendor. Further, all the network nodes are expected to store all published firmware binaries, causing the block size to gradually increase with the transmission of firmware binaries by the vendor time to time [1, 37, 39]. Baza et al. (2018) [38] have proposed a blockchain-based firmware update scheme for autonomous vehicles which employs smart contracts. They used the attribute-based encryption (ABE) technique to set a policy about who has the rights to download and use an update using a smart contract. Their scheme promised to decrease blockchain-based operation by bundling several firmware information in to one transaction using aggregate signature scheme to reduce resource consumption.

Hu et al. (2019) proposed a method based on blockchain, to improve a platform of IoT devices and increase the efficiency and security of its firmware. The method utilized smart contracts to ensure the integrity of the firmware as well as to achieve resistance to malware. They employed peer-to-peer file sharing to ensure the new platform's availability. This method allowed managing different firmware versions belonging to various vendors, while also securing devices against DDoS attacks. Furthermore, their platform was able to check multiple signature requests simultaneously, thus significantly increasing scalability. They also carried out elaborate assessments and simulations to confirm the effective operation of IoT devices on the proposed platform in terms of computing costs and overhead connections [6].

Solomon et al. (2023) proposed a blockchain-based firmware updating method for IoT devices which utilized IPFS to deliver a highly available and distributed encrypted software update binary file. The methodology involves leveraging cryptographic security measures such as Ciphertext-Policy Attribute-Based Encryption (CP-ABE) and hash functions, implementing the framework on the Ethereum blockchain, and conducting simulations on IoT device configurations. The study also includes physical hardware tests on Raspberry Pi devices to validate the proposed framework. This method reduced the computational complexity expected from IoT devices, by offloading such complex operations to the blockchain platform, while also performing authenticity verification, secure delivery and ensuring payment for the software updates. The proposed blockchain-based framework addressed vulnerabilities in current client–server architecture for software update delivery, providing resilience and high availability. The method also significantly reduced the overhead of key generation and communications, improving efficiency and scalability. Through the use of smart contracts, they ensured a seamless delivery without introducing a trusted third party [51].

Another blockchain-based firmware updating method, which is PUSH-based, was introduced by Alexander Yohan, in which new firmware versions from a genuine manufacturer will be delivered. They further used consensus mechanism and smart contract features of blockchain to demonstrate how firmware integrity can be preserved [39]. One issue with this method is that once an update is pushed to the IoT gateway, the update is forwarded from the gateway to devices without any verification. Hence, a chance of malicious code reaching an IoT device is present, and if the gateway has been compromised, the updating process may not be properly executed in this method. In addition, this method could increase network traffic and costs since the vendor is required to release smart contracts every time firmware files or duplicate firmware files are distributed to the broker nodes [1].

However, in situations where a large number of sensors are involved (ex. smart city), use of public blockchain and PoW algorithms is limited in practicality due to their resource intensive nature. This can result in decreased scalability and performance. Moreover, the authors of [18, 37, 38] cases given above do not clearly mention how IoT devices can directly interact with blockchain with their limited computational power. Therefore, it is important to design a scalable, high-performance and lightweight mechanism to ensure IoT device security. Our solution resolves this issue by using DAG-based IOTA-distributed ledger technology and IPFS with lightweight ISO standard MQTT protocol.

3 Materials and methods

3.1 Proposed components and framework

We proposed a system to authenticate the integrity of IoT device firmware and keep it up-to-date, while maintaining secure communication among the three entities: the IoT vendor, IoT gateway and IoT end device, without compromising network performance. Specifically, our approach involves introducing an IoT gateway to establish secure communication between IoT vendors and IoT end devices.

The IoT vendor owns a firmware repository to store firmware binaries and firmware information. MAM client was installed in the vendor's node to initiate a MAM connection. An IPFS cluster was installed for initiating an IPFS connection between the IoT gateways and IoT vendors. Each vendor has its own root address and side key for their restricted MAM channel and IPFS peer identity and swarm key for IPFS connection.

The IoT gateway is connected to the vendor's node via IOTA MAM. The gateway stores information about the connected IoT end device's device type, device Media Access Control (MAC) address, current firmware version, and current firmware hash. Similarly, MAM client was installed in the IoT gateway to initiate a MAM connection with vendor node and IPFS cluster for initiating IPFS connection between other IoT gateways and IoT vendors. The IoT vendor's root key and side key are required in advance to establish the MAM connection and IPFS peer identity and swarm key for IPFS connection. MQTT server is running on IoT gateway; every IoT end device is connected to IoT gateway via MQTT connection over Secure Socket Layer (SSL)/Transport Layer Security (TLS). At the same time, the IoT gateway consists of network time protocol (NTP) service for providing date and time for IoT end devices. There is a separate web interface running on IoT gateway for configuring and monitoring the current status of MAM connection and MQTT connection with IoT end devices. To communicate with IoT end devices, the IoT gateway is equipped with Wi-Fi.

The IoT end device of the proposed system Runs MQTT client over an SSL/TLS connection to the IoT gateway over Wi-Fi.

The proposed scheme operates on two different networks: the IOTA network and the IPFS network (Fig. 1). In our system, IoT vendors and IoT gateways communicate using the MAM protocol by connecting to the IOTA Tangle through a lightweight IOTA full node called 'HORNET'. Each IoT vendor and IoT gateway are connected in the IPFS network by sharing a common swarm key. However, due to hardware limitations of IoT end devices, their integration with IOTA MAM and IPFS is restricted. Therefore, we used the MQTT protocol with SSL/TLS encryption for communication between

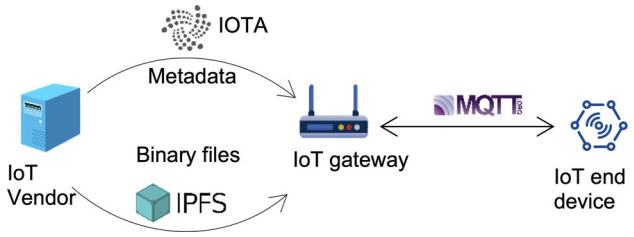

Fig. 1 High-level communication diagram

IoT end devices and the IoT gateway. The function of the proposed scheme is divided into five phases.

Phase 1 - IoT vendor (manufacturer) and IoT gateway registration (Fig. 2)

The publisher uses a secret seed to produce a channel ID and an optional side-key to publish MAM messages to the IOTA Tangle through a full node. Anyone can receive that message from IOTA Tangle using the channel ID. We adopted the restricted mode of MAM to establish a secure connection, in order to allow version control capabilities to the vendor. If a vendor intends to provide access to their firmware information feed, they must share MAM root ID and side-key with the IoT gateway in advance to establish a restricted MAM connection. In return, the IoT gateway retrieves and authenticates the associated data stream from the IOTA Tangle.

IoT devices from different vendors (i.e., different brands) are connected to the IoT gateway, the gateway has to subscribe to each vendor's MAM channel. If the vendor wants to revoke access to their data streams at any time, they can update the authorization key of their MAM channel. After doing so, corresponding subscribers will be unable to decrypt MAM messages. Similarly, pre-configuration is required to establish an IPFS connection between the vendor and the gateway. To create a private IPFS network, we used a swarm key, which serves for authentication and encryption purposes within the private network. Only nodes that possess the correct swarm key can join and communicate within the private IPFS network. Additionally, the IP address of peers and their peer identity are required for reference by all the nodes in this private IPFS network. These details should be shared between IoT vendors and IoT gateways.

Phase 2 - IoT end device registration with IoT gateway (Fig. 3)

Every IoT end device needs to connect to its IoT gateway via Wi-Fi. Therefore, Wi-Fi credentials, MQTT credentials and SSL public key must be shared in advance to establish a secure connection between the IoT gateway and the IoT end device. The IoT end device is configured with a built-in web server with HTML user interface, which can be activated upon the user's request, for example, by pressing a reset button. Upon activation, a 0.96" OLED screen embedded in the device will display a QR code. Scanning this code allows the

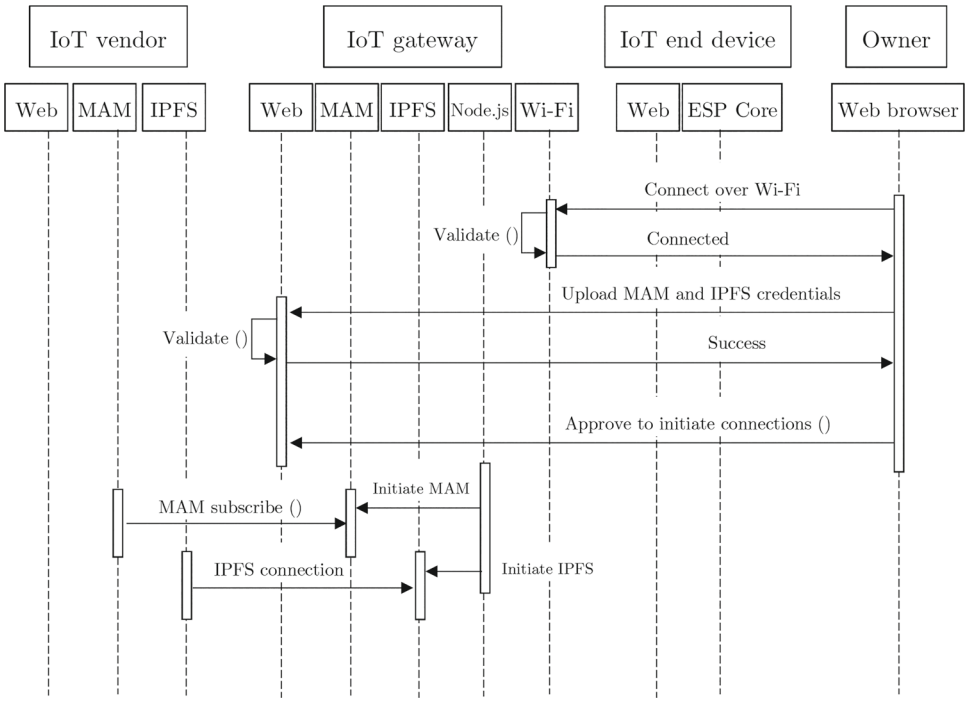

Fig. 2 Diagram for IoT vendor and IoT gateway pre-registration

user to connect to the device's local Wi-Fi hotspot to access the web interface enabling the user to upload these credentials. Once valid credentials are successfully stored in the device's EEPROM, it will connect to the IoT gateway and synchronize the date and time. Then it will listen to the IoT gateway for any firmware update information and continuously broadcast its current firmware version, firmware hash values and hardware version through secure MQTT protocol to the IoT gateway. The IoT gateway will uniquely identify each IoT end device with their Media Access Control (MAC) address.

Phase 3 - New firmware update distribution by the vendor (Fig. 4)

Every vendor node consists of an HTML5-based web interface that runs on Node.js. This portal communicates with the IOTA MAM client and the IPFS client. Firmware deployment is carried out through this web interface. This web portal is capable of uploading firmware binaries to the IPFS network and attaching MAM messages to the IOTA Tangle. When there is a new firmware update to be deployed by the vendor, authorized vendor personnel will access the vendor portal and upload the firmware binaries along with firmware meta-information. The portal then calculates the corresponding SHA256 hash value for the binary file uploaded to the IPFS network and generates an IPFS content ID (CID) to access it via the IPFS network. Once the vendor approves the publication of information to the IOTA Tangle, the portal will include date and time, file size, file hash (SHA256), firmware version, compatible hardware version, device model/type and IPFS CID to a MAM message through the restricted channel. The upload status will be disclosed to the vendor through the web portal.

Phase 4 - IoT gateways receiving firmware information from vendors (Fig. 5)

Each IoT gateway is connected to a single or multiple vendors with their Root ID and side-key and continuously listens to the vendor's update through IOTA Tangle via MAM. When new information is received on the Tangle, it will crosscheck firmware device compatibility with the currently registered IoT end devices. If the Tangle information has the latest firmware information for the local IoT end devices, the IoT gateway will download the firmware files from the IPFS network and hold them in a local temporary location for verification.

Phase 5 - Verification and methods of firmware updating by IoT gateway and IoT end devices (Fig. 6)

Firmware information has been received in advance through IOTA MAM to the IoT gateway. The IoT gateway will run a SHA256-based hash verification for binaries that are downloaded and stored locally in Phase 4. If the hash verification is successful, it will move firmware files to the local repository and push the latest firmware information to the IoT devices.

All the IoT end devices, which are connected (online) to the IoT gateway at the time, will receive a message for immediate firmware update. This process is called force validation (Method 1) (Fig. 7). IoT end devices that are offline or not connected will be able to receive new firmware information once the device is registered with the IoT gateway and becomes online. Every IoT end device is configured to

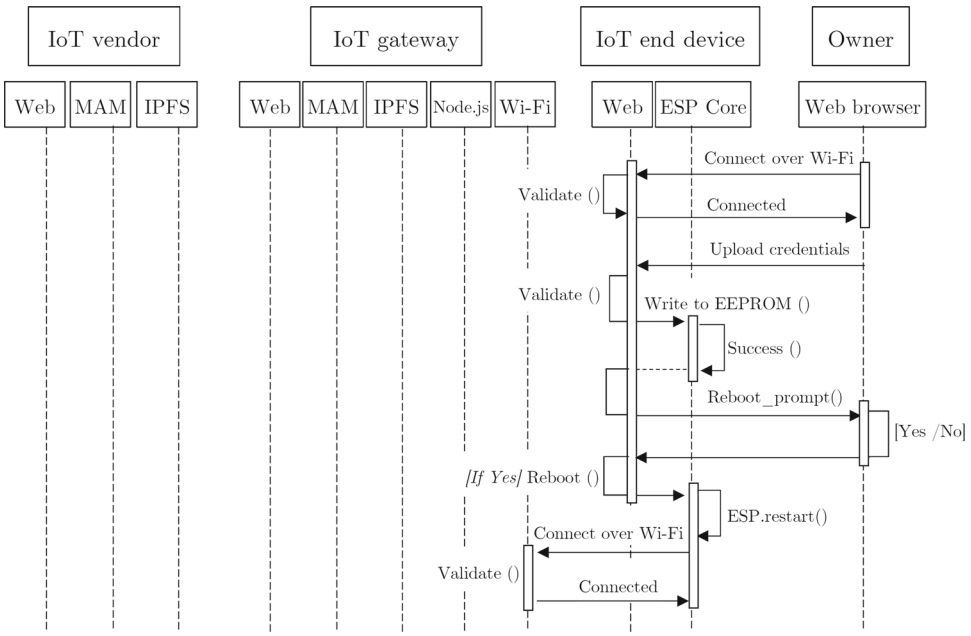

Fig. 3 Diagram for configuring a new IoT end device to IoT gateway

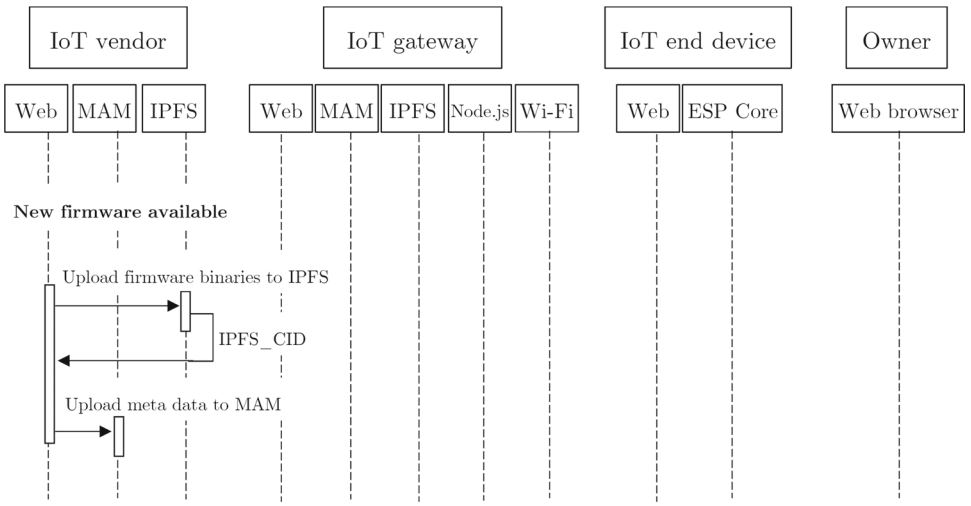

Fig. 4 Diagram for new firmware update deploy by IoT vendor

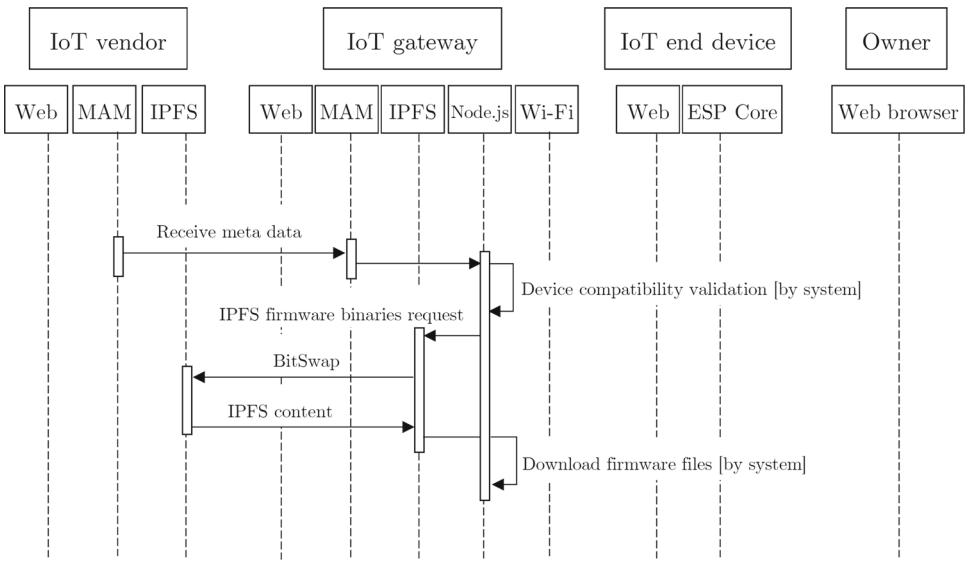

Fig. 5 Diagram of IoT gateway receiving meta data from IOTA MAM and retrieving firmware binaries from IPFS

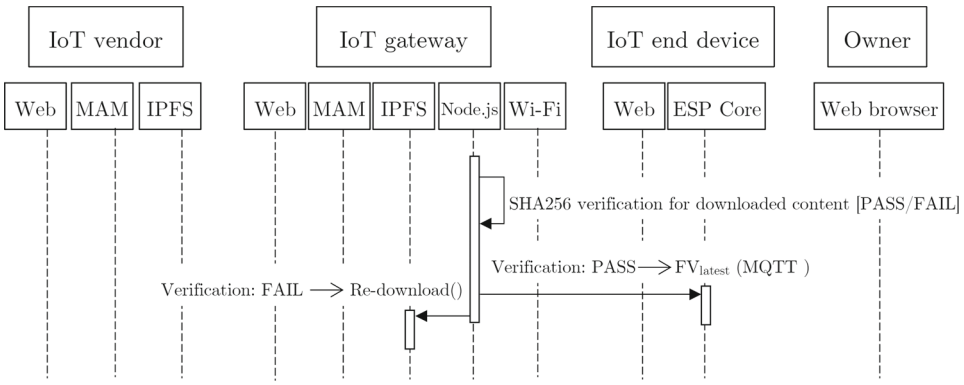

Fig. 6 Diagram for IoT gateway performing hash verification for firmware content

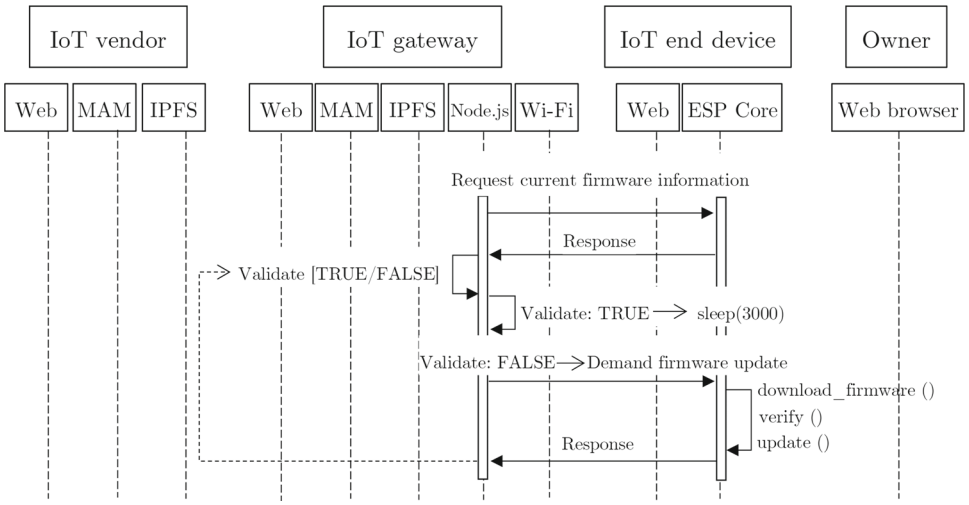

Fig. 7 Method 1: Force validation - IoT gateway will validate IoT device firmware periodically

self-check latest firmware information from the IoT gateway every five seconds. This process is called self-validation (Method 2) (Fig. 8). IoT devices will access the IoT gateway's firmware repository to check the latest firmware information. If the IoT device detects new firmware, it will run a hash verification. When the firmware binaries are authenticated, it will upgrade its firmware and then reboot with the latest update applied.

3.2 Implementation

We built a proof-of-concept prototype using two Linux Raspberry Pi 3B+ as IoT gateway and vendor's node (Fig. 9). These two nodes' operating systems (OS) are Ubuntu 20.04 LTS (Focal Fossa) ARM 64bit. On top of the OS, we have configured HORNET, an IPFS private cluster and a MQTT server with SSL/TLS. The implementation of the IoT end device was done by using an ESP8266 and ESP32-based development kit. This board was chosen considering its low-cost, smaller size, WiFi capability, and C-programming support using the Arduino development tool. ESP8266 has a 4 MB flash, while ESP32 has a 16MB flash. The 0.96" 128x64 OLED display was used with IoT end device for debugging purposes.

Smart home IoT gateway

This device is implemented to bridge the gap between IoT vendor and IoT end device. This device will consistently maintain communication with the IoT vendor to receive the latest firmware updates and their metadata, which also include the IPFS access credentials. Communication with vendors is achieved through IOTA MAM, to verify the firmware information, convert that information to MQTT messages and send it to the IoT end devices. The vendor's channel ID and secret key will be pre-shared with the IoT gateways.

IoT vendor

An IoT vendor was implemented to simulate the deployment of firmware updates for IoT end devices. This device will consistently maintain communication with the Smart home IoT gateway using IOTA and IPFS. An HTML5-based web interface was designed as a portal for authorized administrators to deploy firmware updates.

IoT end device

The IoT end device communicates directly with the IoT gateway via Wi-Fi connection. MQTT communication over SSL/TLS is used as the communication protocol. Each device is equipped with an embedded web server running, which can be toggled on when required for a configuration. Configura-

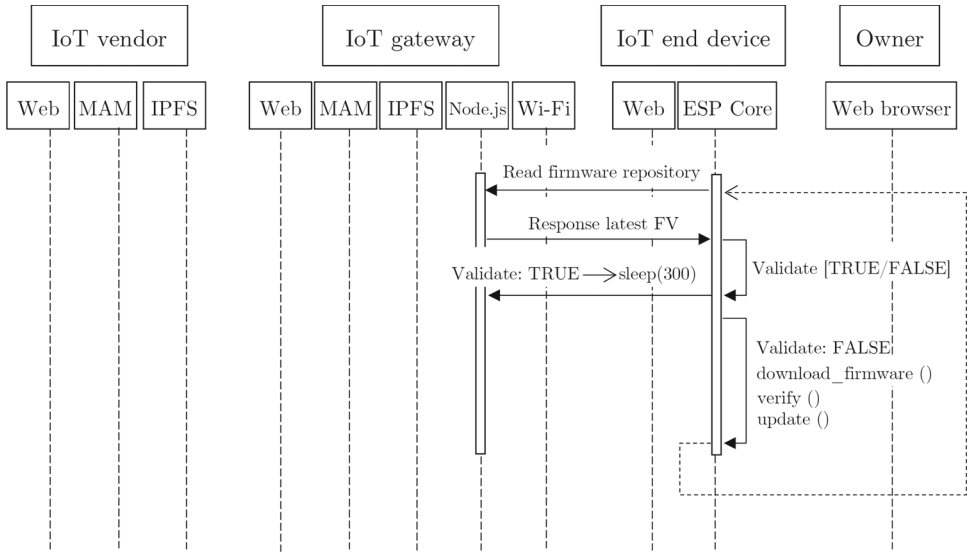

Fig. 8 Method 2: Self validation - IoT device will periodically check latest firmware from IoT gateway

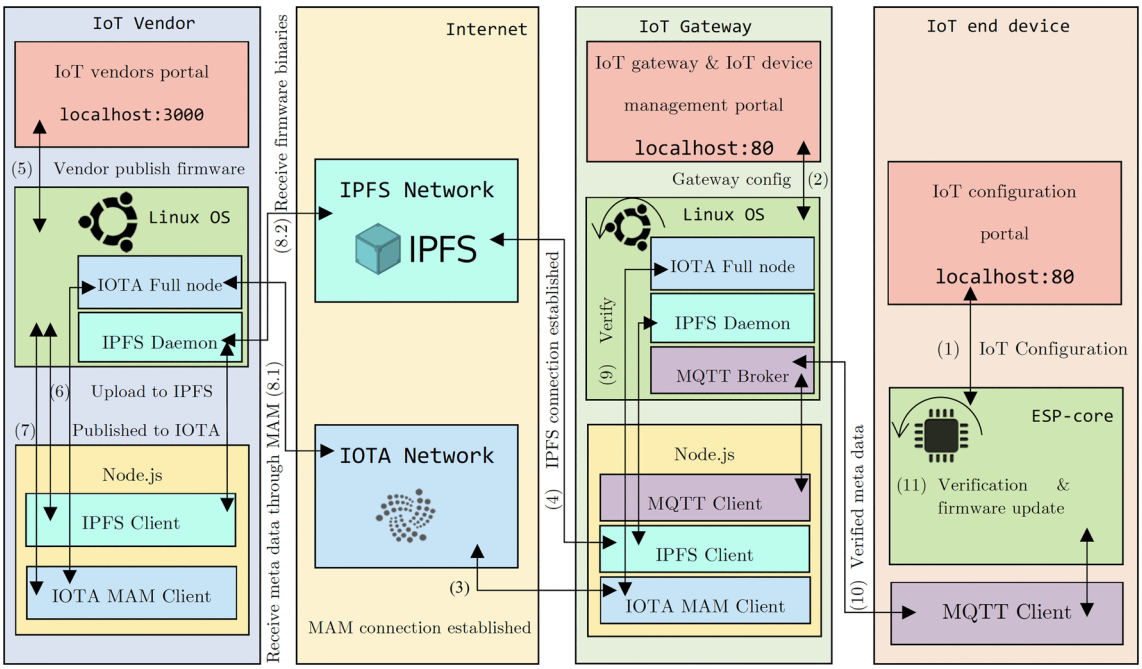

Fig. 9 End-to-end implementation overview

tion can be activated by pressing a button while turning the device on.

Quantification of communication latency

We measured the system's latency between an IoT vendor and IoT gateway's MAM communication for a given firmware update. We ran five sets of 300 latency tests each consisting of payloads ranging from 400 to 600 characters. The average latency for each payload was calculated in seconds for each set. Performance of the current study was compared to previous studies through a feature comparison.

4 Results and discussion

The results of the latency tests are given in Fig. 10. We found that the average time required to complete a MAM communication between an IoT vendor and an IoT gateway is approximately 32 s for all tested payloads. In addition, we evaluated the performance of the proposed system in terms of authenticity, scalability, latency and transactions fees with previous studies, as summarized in Table 1. In standard firmware update methods used for many IoT devices, some vendors provide validation information with their repositories, such as hash values to authenticate their binaries.

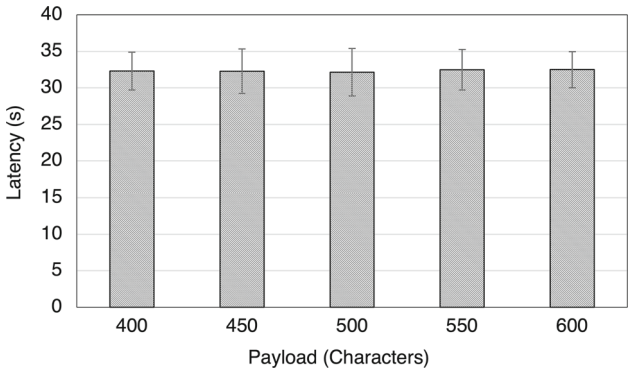

Fig. 10 Variation of latency with payload

However, that process is not scalable due to the scalability limitations of their communication protocols. Compared to blockchain-based firmware update methods by Lee et al. (2017) [18], Boudguiga et al. (2017) [37], Hu et al. (2019)[6] and Solomon et al. (2023) [51], our method performs IOTA MAM-based firmware meta data communication, IPFS-based decentralized firmware file distribution, a hash verification for firmware binaries and autonomous firmware update mechanisms. Due to the application of IOTA, IPFS and MQTT where scalability comes along with their technologies, our method is highly scalable.

5 Conclusions

In order to address security, cost and latency issues associated with previous studies, we developed an IoT gateway utilizing the IOTA MAM protocol and the IPFS-distributed file system. To the extent of our knowledge, this is the first work which uses IOTA MAM for firmware updating and authentication.

We implemented a proof of concept and tested against IOTA Tangle to authenticate the integrity and authenticity of IoT device firmware and to keep it up-to-date using the proposed scheme. Our developed scheme securely shares the latest firmware content over IOTA and IPFS networks and autonomously performs secure firmware update in IoT end devices to ensure the authenticity and firmware integrity. We implemented system infrastructure of the IoT gateway and the IoT vendor using two Raspberry Pi 3B+ devices with Ubuntu 20.04 LTS ARM 64bit run on IOTA full nodes, Node.js, IOTA MAM client libraries, IPFS cluster configuration and secure MQTT server. We established communication between the IoT gateway and the IoT end device over Wi-Fi, secured with SSL/TLS using MQTT protocol. In addition, we developed a web interface to configure and monitor the connectivity between each entity. The IoT end device was implemented using C++ on ESP8266 and ESP32 Wi-Fi microcontrollers.

Compared to past research [6, 18, 37, 51], our method performs a hash verification for firmware binaries and an autonomous firmware update mechanism that sends a message through the IOTA tangle. Furthermore, our method is highly scalable, has a low latency and charges no transaction fees, as we use IOTA protocol instead of blockchain technology.

The proposed system can be further enhanced for healthcare data applications while adhering to privacy regulations and data processing best practices. By carefully considering the use case and designing the data exchange system, we can

Table 1 Feature comparison with previous studies

	Lee et al. [18]	Boudguiga et al. [37]	Hu et al. [6]	Solomon et al. [51]	Proposed
Currency	Bitcoin	Bitcoin	Ethereum	N/A	IOTA
DLT Architecture	Blockchain	Blockchain	Blockchain	Blockchain	Directed acyclic graph
Performance	$\propto$ 1/N	$\propto$ 1/N	$\propto$ 1/N	$\propto$ 1/N	$\propto$ N
Smart Contract	No	No	Yes	Yes	No
Firmware file authentication	No	No	No	No	Yes (SHA256)
Firmware file availability	High	Low	High	High	High
Firmware update mechanism	Pull	Pull	Pull and Push	Pull	Pull and Push
IPFS-based firmware sharing	No	No	No	Yes	Yes
IoT computation complexity	High	High	Medium	Medium	Low
IoT device management	No	No	No	No	Yes
Support multiple vendors	No	Yes	Yes	Yes	Yes

N/A - Not available, N - Number of transactions

address the potential conflicts between DLT and data privacy regulations.

Code and data

Our implementation source code is available at https://www.github.anushkawijesundara.com.

Declarations

conflict of interest The authors declare no conflict of interest. The authors did not receive support from any organization for the submitted work.

References

1. Choi, S., Lee, J.H.: Blockchain-based distributed firmware update architecture for IoT devices. IEEE Access **8**, 37518 (2020). https://doi.org/10.1109/ACCESS.2020.2975920
2. Rastegari, H., Nadi, F., Lam, S.S., Ikhwanuddin, M., Kasan, N.A., Rahmat, R.F., Mahari, W.A.W.: Internet of Things in aquaculture: a review of the challenges and potential solutions based on current and future trends. Smart Agric. Technol. **4**, 100187 (2023). https://doi.org/10.1016/j.atech.2023.100187
3. Brogan, J., Baskaran, I., Ramachandran, N.: Authenticating health activity data using distributed ledger technologies. Comput. Struct. Biotechnol. J. **16**, 257 (2018). https://doi.org/10.1016/j.csbj.2018.06.004
4. Al-Habaibeh, A., Yaseen, S., Nweke, B.: A comparative study of low and high resolution infrared cameras for IoT smart city applications. Ain Shams Eng. J. **14**, 102108 (2023). https://doi.org/10.1016/j.asej.2022.102108
5. Liu, Y.N., Wang, Y.P., Wang, X.F., Xia, Z., Xu, J.F.: Privacy-preserving raw data collection without a trusted authority for IoT. Comput. Netw. **148**, 340 (2019). https://doi.org/10.1016/j.comnet.2018.11.028
6. Hu, J.W., Yeh, L.Y., Liao, S.W., Yang, C.S.: Autonomous and malware-proof blockchain-based firmware update platform with efficient batch verification for Internet of Things devices. Comput. Secur. **86**, 238 (2019). https://doi.org/10.1016/j.cose.2019.06.008
7. Nadir, I., Mahmood, H., Asadullah, G.: A taxonomy of IoT firmware security and principal firmware analysis techniques. Int. J. Crit. Infrastruct. Prot. **38**, 100552 (2022). https://doi.org/10.1016/j.ijcip.2022.100552
8. Jaouhari, S.E., Bouvet, E.: Secure firmware over-the-air updates for IoT: survey, challenges, and discussions. Internet of Things **18**, 100508 (2022). https://doi.org/10.1016/j.iot.2022.100508
9. Yohan, A., Lo, N.W., Achawapong, S.: Blockchain-based firmware update framework for internet-of-things environment (2018)
10. Xu, Q., Aung, K.M.M., Zhu, Y., Yong, K.L.: A blockchain-based storage system for data analytics in the internet of things (2018). https://doi.org/10.1007/978-3-319-58190-3_8
11. Tan, C.J., Mohamad-Saleh, J., Zain, K.A.M., Aziz, Z.A.A.: ACM, pp. 186–190 (2017). https://doi.org/10.1145/3132300.3132337
12. Kim, J., Chou, P.H.: Energy-efficient progressive remote update for flash-based firmware of networked embedded systems. ACM Trans. Des. Autom. Electron. Syst. **16**, 1 (2010). https://doi.org/10.1145/1870109.1870116
13. Khan, M.A., Salah, K.: IoT security: review, blockchain solutions, and open challenges. Futur. Gener. Comput. Syst. **82**, 395 (2018). https://doi.org/10.1016/j.future.2017.11.022
14. Choi, B.C., Lee, S.H., Na, J.C., Lee, J.H.: Secure firmware validation and update for consumer devices in home networking. IEEE Trans. Consum. Electron. **62**, 39 (2016). https://doi.org/10.1109/TCE.2016.7448561
15. Symantec. Internet security threat report (2019)
16. Sarfraz, U., Alam, M., Zeadally, S., Khan, A.: Privacy aware IOTA ledger: decentralized mixing and unlinkable IOTA transactions. Comput. Netw. **148**, 361 (2019). https://doi.org/10.1016/j.comnet.2018.11.019
17. Wijesundara, A., Joong-Sun, L., Tith, D., Suzuki, H., Obi, T.: Development of a Firmware Authenticating and Updating Scheme for Smart Home IoT Devices Using Distributed Ledger Technologies. (Computer Security Symposium 2019 (IPSJ), 2019), pp. 817–823
18. Lee, B., Lee, J.H.: Blockchain-based secure firmware update for embedded devices in an Internet of Things environment. J. Supercomput. **73**, 1152 (2017). https://doi.org/10.1007/s11227-016-1870-0
19. Khelif, M.A., Lorandel, J., Romain, O., Regnery, M., Baheux, D.: A versatile emulator of MitM for the identification of vulnerabilities of IoT devices, a case of study. (ACM, 2019), pp. 1–6. https://doi.org/10.1145/3341325.3342019
20. Sousa, B.F.L.M., Abdelouahab, Z., Lopes, D.C.P., Soeiro, N.C., Ribeiro, W.F.: An intrusion detection system for denial of service attack detection in internet of things. (ACM, 2017), pp. 1–8. https://doi.org/10.1145/3018896.3018962
21. Moran, B., Tschofenig, H., Brown, D., Meriac, M.: A firmware update architecture for internet of things (2021). https://doi.org/10.17487/RFC9019
22. Kolokotronis, N., Limniotis, K., Shiaeles, S., Griffiths, R.: Secured by blockchain: safeguarding internet of things devices. IEEE Consum. Electron. Mag. **8**, 28 (2019). https://doi.org/10.1109/MCE.2019.2892221
23. Zandberg, K., Schleiser, K., Acosta, F., Tschofenig, H., Baccelli, E.: Secure firmware updates for constrained IoT devices using open standards: a reality check. IEEE Access **7**, 71907 (2019). https://doi.org/10.1109/ACCESS.2019.2919760
24. Zhao, Y., Liu, Y., Tian, A., Yu, Y., Du, X.: Blockchain based privacy-preserving software updates with proof-of-delivery for Internet of Things. J. Parallel Distrib. Comput. **132**, 141 (2019). https://doi.org/10.1016/j.jpdc.2019.06.001
25. Huh, S., Cho, S., Kim, S.: Managing IoT devices using blockchain platform. (IEEE, 2017), pp. 464–467. https://doi.org/10.23919/ICACT.2017.7890132
26. Fernandez-Carames, T.M., Fraga-Lamas, P.: A review on the use of blockchain for the Internet of Things. IEEE Access **6**, 32979 (2018). https://doi.org/10.1109/ACCESS.2018.2842685
27. Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L.,

Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K., Zhou, Y.: Understanding the mirai botnet. (USENIX Association, 2017), pp. 1093–1110
28. Leiba, O., Bitton, R., Yitzchak, Y., Nadler, A., Kashi, D., Shabtai, A.: IoTPatchPool: incentivized delivery network of IoT software updates based on proofs-of-distribution. Pervasive Mob. Comput. **58**, 101019 (2019). https://doi.org/10.1016/j.pmcj.2019.04.010
29. Nizamuddin, N., Salah, K., Azad, M.A., Arshad, J., Rehman, M.: Decentralized document version control using ethereum blockchain and IPFS. Comput. Electr. Eng. **76**, 183 (2019). https://doi.org/10.1016/j.compeleceng.2019.03.014
30. Pillai, A., Sindhu, M., Lakshmy, K.: Securing firmware in internet of things using blockchain. (IEEE, 2019), pp. 329–334. https://doi.org/10.1109/ICACCS.2019.8728389
31. Mtetwa, N., Tarwireyi, P., Adigun, M.: Secure the internet of things software updates with ethereum blockchain. (IEEE, 2019), pp. 1–6. https://doi.org/10.1109/IMITEC45504.2019.9015865
32. Pierro, M.D.: What Is the blockchain? Comput. Sci. Eng. **19**, 92 (2017). https://doi.org/10.1109/MCSE.2017.3421554
33. Dorri, A., Kanhere, S.S., Jurdak, R.: Blockchain in internet of things: challenges and solutions (2016)
34. Saad, M., Njilla, L., Kamhoua, C., Kim, J., Nyang, D., Mohaisen, A.: Mempool optimization for defending against DDoS attacks in PoW-based blockchain systems. (IEEE, 2019), pp. 285–292. https://doi.org/10.1109/BLOC.2019.8751476
35. Vujicic, D., Jagodic, D., Randic, S.: Blockchain technology, bitcoin, and Ethereum: a brief overview. (IEEE, 2018), pp. 1–6. https://doi.org/10.1109/INFOTEH.2018.8345547
36. Kawase, Y., Kasahara, S.: Transaction-confirmation time for bitcoin: a queueing analytical approach to blockchain mechanism (2017). https://doi.org/10.1007/978-3-319-68520-5_5
37. Boudguiga, A., Bouzerna, N., Granboulan, L., Olivereau, A., Quesnel, F., Roger, A., Sirdey, R.: Towards better availability and accountability for IoT updates by means of a blockchain. (IEEE, 2017), pp. 50–58. https://doi.org/10.1109/EuroSPW.2017.50
38. Baza, M., Nabil, M., Lasla, N., Fidan, K., Mahmoud, M., Abdallah, M.: Blockchain-based firmware update scheme tailored for autonomous vehicles. (IEEE, 2019), pp. 1–7. https://doi.org/10.1109/WCNC.2019.8885769
39. Yohan, A., Lo, N.W.: FOTB: a secure blockchain-based firmware update framework for IoT environment. Int. J. Inf. Secur. **19**, 257 (2020). https://doi.org/10.1007/s10207-019-00467-6
40. Raschendorfer, A., Mörzinger, B., Steinberger, E., Pelzmann, P., Oswald, R., Stadler, M., Bleicher, F.: On IOTA as a potential enabler for an M2M economy in manufacturing. Procedia CIRP **79**, 379 (2019). https://doi.org/10.1016/j.procir.2019.02.096
41. Babich, V., Hilary, G.: Blockchain and other distributed ledger technologies in operations. foundations and trends® in technology, information and operations management **12**, 152 (2019). https://doi.org/10.1561/0200000084
42. Pervez, H., Muneeb, M., Irfan, M.U., Haq, I.U.: A comparative analysis of DAG-based blockchain architectures. (IEEE, 2018), pp. 27–34. https://doi.org/10.1109/ICOSST.2018.8632193
43. Benet, J.: Ipfs - content addressed, versioned, p2p file system (2014)
44. Popov, S., Saa, O., Finardi, P.: Equilibria in the tangle. Comput. Ind. Eng. **136**, 160 (2019). https://doi.org/10.1016/j.cie.2019.07.025
45. Pinjala, S.K., Sivalingam, K.M.: DCACI: a decentralized lightweight capability based access control framework using IOTA for internet of things. (IEEE, 2019), pp. 13–18. https://doi.org/10.1109/WF-IoT.2019.8767356
46. IOTA. Transactions | getting started | iota documentation (2020). https://docs.iota.org/docs/getting-started/1.0/introduction/overview
47. Akbulut, S., Semantha, F.H., Azam, S., Pilares, I.C.A., Jonkman, M., Yeo, K.C., Shanmugam, B.: Designing a private and secure personal health records access management system: a solution based on IOTA distributed ledger technology. Sensors **23**, 5174 (2023). https://doi.org/10.3390/s23115174
48. Gangwani, P., Perez-Pons, A., Joshi, S., Upadhyay, H., Lagos, L.: Integration of data science and IoT with blockchain for industry **4**,(2023). https://doi.org/10.1007/978-981-19-8730-4_6
49. Ali, M.S., Dolui, K., Antonelli, F.: IoT data privacy via blockchains and IPFS. In: Proceedings of the Seventh International Conference on the Internet of Things. (ACM, 2017), pp. 1–7. https://doi.org/10.1145/3131542.3131563
50. Hawig, D., Zhou, C., Fuhrhop, S., Fialho, A.S., Ramachandran, N.: Designing a distributed ledger technology system for interoperable and general data protection regulation-compliant health data exchange: a use case in blood glucose data. J. Med. Internet Res. **21**, e13665 (2019). https://doi.org/10.2196/13665
51. Solomon, G., Zhang, P., Brooks, R., Liu, Y.: A secure and cost-efficient blockchain facilitated IoT software update framework. IEEE Access **11**, 44879 (2023). https://doi.org/10.1109/ACCESS.2023.3272899