Version Control Using Distributed Ledger Technologies for Internet of Things Device Software Updates

Magnus Westerlund
Department of Business Management and Analytics
Arcada University of Applied Sciences
Helsinki, Finland
magnus.westerlund@arcada.fi

John Wickström
Department of Business Management and Analytics
Arcada University of Applied Sciences
Helsinki, Finland
wickstjo@arcada.fi

Göran Pulkkis
Department of Business Management and Analytics
Arcada University of Applied Sciences
Helsinki, Finland
goran.pulkkis@arcada.fi

Abstract—**The number of installed Internet of Things (IoT) devices is growing rapidly and securing these IoT installations is an important task that may require technical knowledge that the owners of these devices do not always possess. Although experts have pointed out, that security should always be a priority when creating IoT products, the challenges are numerous and security solutions are not always targeted to decentralized or distributed architectures. In this paper, we explore the mechanisms for creating a method for a distributed IoT software update service that utilize distributed ledger technologies, such as Ethereum smart contracts and the InterPlanetary File System (IPFS). Our aim is to present a method that offers a more transparent version control of updates than current solutions, which are mostly conceptually centralized. We also aim to avoid relying on a central node for distributing updates and to create a fully secured and automated process for update management.**

Keywords-IoT; distributed ledger; blockchain; version control; software update.

I. INTRODUCTION

Version control has been an integral part of software development for a long time. Common techniques and methods for provisioning IT-services (incl. configuration, deployment, orchestration and management) depend on formalizing a process for handling changes made to files and data. Version Control Systems (VCS) became commonplace in the late 1990s and initially catered mostly to intra-organizational software development (internally) and a well-known system was Rational ClearCase [1]. As software development matured and inter-organizational development (between organizations), became commonplace through, e.g., open source development, new distributed VCS, such as Git [2], emerged. These VCS have distributed features primarily from the perspective of access, who can collaborate and contribute to a project hosted on a web-based Git repository (such as Gitlab). Although it is possible to mirror Git repositories, these services have no proper distributed features in terms of inherent trustless consensus and guarantee for service availability. Such fears among developers were quite evident when Microsoft acquired GitHub [3], another Git-based repository many open source projects are relying on.

Traditionally, version control has strictly meant tracking changes in text-based files. To store binary files in a VCS offers mainly a stored version path. For some binary files, plugins exist that will allow a diff to executed, but often this would be an exception. However, there are new use cases for version control that go beyond the initial ability of performing a comparison between file versions. These use cases are coming from new technologies, such as machine learning (incl. Artificial Intelligence (AI)) and Internet of Things (IoT). For an AI-enabled service, version control extends to, that the process must include training data, network initialization, parameter settings, and serialization of the trained network to a file. Often, other types of metadata should be stored as well, e.g., statistical properties of training data, output quality metrics and naturally if the model is updated online it requires further measures. Any autonomous AI-based service aimed for production use will need continuous catering for forensic investigations during the longevity of the service.

This paper focuses on the IoT use case, to extend the understanding for what purposes version control is usable and how to implement a Proof of Concept (PoC) of a VCS for IoT software updates using Distributed Ledger Technology (DLT), such as blockchain, smart contracts, and the InterPlanetary File System (IPFS) [4]. This paper adopts a methodology intended to identify single case mechanisms through an exploratory approach [5]. Our long-term research aim is to develop a new methodology for fully secured and automated IoT device updates. This process must also be transparent in terms of who has created updates and be auditable in case there are detected vulnerabilities. We limit the scope for this paper to the backend architecture utilizing IPFS, Ethereum smart contracts, and browser-based Distributed Applications (DApp).

This paper has the following structure: Section II describes the problem setting of reliability in IoT devices. Section III presents relevant IoT policies and standardization efforts. Section IV discusses DLT-based update services. Section V surveys other proposals for distributed update services and presents our PoC. Section VI contains conclusion and proposal for future development

II. PROBLEM SETTING FOR IOT DEVICE RELIABILITY

The proliferation of IoT devices and services based on these are helping to digitize the physical landscape. IoT enabled devices have been introduced into almost any setting and convey large volume of data and varieties of data, e.g., in the format of video, sound, and potentially any data type that can be measured with a sensor that converts analogue measurements into a digital data flow. We can anticipate the technological progress will continue to shape new domains in our lives and within the coming decades, extending to include many new areas, e.g., personal healthcare and home automation. These new domains will introduce a myriad of highly sensitive information sources, information that must be processed, and often stored for an indefinite and sometimes an infinite period for the digitization of these areas. By embedding information-sharing electronics into everyday physical objects, we will create a “global cyber-physical infrastructure” [6]. IoT uses standardized communication protocols and merges computer networks into a “common global IT platform of seamless networks and networked “Smart things/objects”” [7]. From the perspective of platform and service innovation, by utilizing IoT technology, the focus will be on creating AI-enabled services that are able to draw inferences from the data collected from IoT devices. This will offer users descriptive answers, predict future behavior and needs, and eventually provide prescriptive suggestions for improving daily life. We here define AI-enabled services as based on machine learning techniques that infer decision support or decisions based on the collected IoT data. Therefore, relying on data veracity becomes crucial for the trustworthiness of these services.

Network and information security are often more challenging for IoT systems than for traditional networks. Cloud resources used by many IoT systems are publicly accessible and thereby, through this availability, increase the risk of intrusion. The increase in the processing of sensitive data in IoT systems makes security challenges more noteworthy, particularly in light of legal issues around cross-border transfers and data protection [8]. The debate regarding a sustainability problem in IoT security has resulted in some experts calling for a halt to IoT deployments and innovations [9] and that IoT devices should come with public safety warnings [10]. This paper takes the position that there is currently a sustainability problem in IoT security and we should innovatively address this problem with new secure IoT management methods designed specifically for the distributed architecture of IoT networks.

III. CURRENT IOT POLICY SITUATION

In a traditional IoT architecture, IoT devices are network nodes, which transmit their data (incl. logs) to a data store through some proxy. IoT device management for enterprise-level devices is often a manual process, whereas consumer devices may query a manufacturer-defined end-point for software updates, which typically are impossible to validate for origin or content. Device administrators have local credentials for authentication, but an Identity and Access Management (IAM) solution is often missing. Only authenticated users should have authorization to access IoT devices and to update device firmware from device deliverers’ databases. A system log stored on a respective node would require device access for collection (pull) of data. Storage space is often very limited so only the most recent activities may be stored on the device. Hence, continuous collection to an external data store is required.

From an accountability perspective, continuous delivery of new updates to a node is also a necessity, something that often requires a manual process by a system administrator. The manufacturer should also provide new software (e.g., firmware) security updates for the lifetime of said IoT devices. For this process to be complete, traditional IoT systems require many manual process steps that are often not possible to ensure in today’s environment. Hence, we find it motivated to propose a new type of architecture better suited to a decentralized or distributed network topology.

A secure IoT system is one that can fulfil the following criteria [11]:

- does not contain any hardware, software, or firmware component with any known security vulnerabilities or defects,
- relies on software or firmware components capable of accepting properly authenticated and trusted updates from the vendor,
- uses only non-deprecated industry-standard protocols and technologies for functions such as communication, encryption, and intercommunication with other devices, and
- does not include any fixed or hard-coded credentials used for remote administration, the delivery of updates, or communication.

The Cloud Security Alliance (CSA) [12] IoT Working Group published in 2018, 10 security recommendations for IoT firmware updates [13]. The recommendations focus on device integrity and the use of a conceptually centralized service backend.

A. IoT Device Software Update Standardization Efforts

The Internet Engineering Task Force (IETF) has a currently active Security Area Working Group called Software Updates for Internet of Things (suit) [14]. The focus is on secure firmware update solutions, which include a mechanism for transporting firmware images to compatible devices, a digitally signed manifest containing firmware image meta-data and the firmware image(s).

A recent informational Internet-Draft [15] defines that the purpose of an IoT firmware update is to fix vulnerabilities, to update configuration settings, and to add new functionality for improved security. A firmware update must ensure firmware image authentication and integrity protection. In certain cases, prevention of the use of modified firmware images or images from unknown sources may be necessary. Here, it is important to understand the dilemma of potentially installing vulnerable software, versus an informed operator installing a trusted open source-based alternative. Encryption based confidentiality protection can prevent unauthorized access to and modification of the plaintext binary of a firmware image. However, encryption may decrease transparency in some cases.

Firmware updates can be client-initiated by polling for new firmware images or server-initiated by status tracking of IoT devices. A firmware update in an IoT device consists of following steps [15]:

- the device is notified that an update exists,
- a pre-authorization verifies if the manifest signer is authorized to update device firmware. IoT device decide on acceptance of the new firmware image,
- dependency resolution is needed when more than one firmware component can be updated,
- a local copy of the firmware image is downloaded,
- the image is processed into a format the IoT device can recognize and install. Thereafter, the bootloader boots from the installed firmware image.

IV. Distributed update Services Based on DLT Concepts

As discussed in the previous section, traditional IT-architectures, incl. cloud computing based Software-as-a-Service, rely mainly on a conceptually centralized service provision model, while IoT networks and DLT originate from decentralized or distributed architectures. The Bitcoin blockchain [16] introduced a cryptographically secured and distributed ledger. The ability to append transactions to an otherwise immutable ledger comes from a distributed and pseudonymous consensus mechanism, i.e., Nakamoto consensus [16]. Bitcoin's consensus protocol includes both a validity check of a certain transaction and an information sharing protocol, where accepted transactions are stored in blocks chained together in a chronological order. The ledger is an immutable transactional database, thus, the blockchain only stores transactional changes and thereby stays immutable by not forcing an update on pre-existing variable values. In [17], this represents the first generation of DLT. The second DLT generation is in [17] defined to be based on smart contracts, which not only perform an authentication of users and verification of transactions, but may also involve more advanced logical condition states for authorization and automated continuous verification of these condition states.

DLT-based protocol extensions to the web software stack have inter alia provided a new distributed approach to provisioning web services. IPFS provides a Peer-to-Peer (P2P) hypermedia protocol [4] that makes it possible to distribute high volumes of data with high efficiency. IPFS is a distributed file system that utilizes content addressing to fetch static information, rather than location addressing like most traditional file systems. Hashing the content of files or the entire directories achieves this. A resulting hash string works as a link, which also makes IPFS immune to duplicate files. IPFS file versioning based on the generated content identifier (hash) is directly usable for a known latest revision. However, an InterPlanetary Name System (IPNS) [18] identifier exists based on the node peer ID that provides a mutable resource link to the IPFS file hash which when published can be bound to the IPNS. Accessing a file through the IPNS link allows the revision of the IPFS file to change, by republishing the new hash of the file to the IPNS; see Fig. 1 for an illustration.

The data structure behind IPFS is the Merkle Directed Acyclic Graph (DAG), whose links are hashes. Users are the individual peer nodes in a larger swarm. All hashed content published in that particular swarm is retrievable for any participating user. Each IPFS node utilizes a Public Key Infrastructure (PKI) based identification that generates an IPNS, which is a self-certifying PKI namespace (IPNS). This provides all objects in IPFS some useful properties:

- authenticated content,
- permanent cached content,
- a universal data structure, a Merkle DAG,
- a decentralized platform for collaboration.

V. Version Control DLT Solutions

This paper focuses on a new extended VCS use case for fully secured and automated IoT device updates and related management. The first sub-section presents the existing literature of other proposed DLT-based solutions to version control of IoT software updates. The following sub-sections presents initial results of our study on how to implement a fully secured and automated architecture for handling IoT software updates. As stated, the aim of this research is to bring transparency into the process of maintaining IoT devices, by utilizing the earlier mentioned beneficial properties of IPFS and smart contracts executed on the Ethereum blockchain and the Ethereum virtual machine. The benefit of these distributed ledger technologies is that they are, often similarly to the architecture for IoT devices, based on an automated process and can be configured to construct a decentralized platform. Therefore, combining these technologies in a system architecture should improve the reliability, maintainability, and forensic abilities in IoT network supervision.

A. Proposed DLT-Based Solutions to Version Control of IoT Software Updates.

Several authors have accentuated the data structure similarities between Git and DLT, and that it is possibly usable for some form of DLT-based version control, e.g., "blockchain can be seen as a Peer-to-Peer (P2P) hosted Git repository" [19].

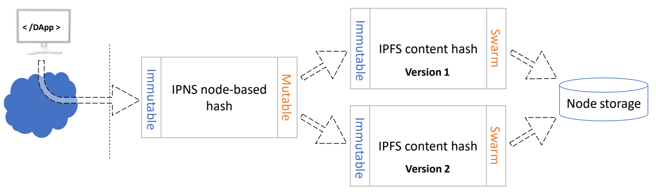

Figure 1. IPFS addressing process flow.

Feature wise Git repository branches are blockchains according to the definition “A blockchain is a sequence of blocks of data in which each block, other than the first, is cryptographically linked to its predecessor” [20]. A blockchain network definition is “a Peer-to-Peer network in which peers collaborate to achieve a common goal by using a blockchain” [20]. According to this definition, a Git repository is a blockchain network, the Git repository peers are developers in a software development project, and the blockchain is the master branch. Git peers often fork the master branch when new versions are stored in the repository, collaborate on the master branch, and strive to merge other branches with the master branch. A Git repository is permission-based and consensus is trust-based on some Git hosting service.

In a proposed setup for IoT device firmware updates, a device manufacturer provides a master update node and configures all IoT devices from the manufacturer as nodes in the same blockchain network [21]. The setup deploys a smart contract for storing the hash of the latest firmware update as a transaction record in a blockchain and for retrieving the latest stored transaction record. The corresponding firmware file is stored in a distributed P2P file system such as IPFS. The manufacturer’s blockchain node stores new firmware updates. IoT devices can find hashes of new firmware updates by querying the smart contract and then to request and locally store the firmware file from the distributed P2P file system by its hash. An IoT device joining the blockchain network after the manufacturer’s node has left the network can therefore still retrieve the latest firmware update. The hash stored of the blockchain verifies that the firmware file stored in a distributed P2P file system is untampered

For another proposed blockchain-based solution for secure firmware updates in IoT devices [22], the blockchain network consists of normal nodes, which are IoT devices and verification nodes, which store firmware files and hash values of firmware files called verifiers in their databases. Outside the blockchain network are firmware vendor nodes. A vendor node maintains a secure communication channel to a verification node for delivery of new firmware updates. A normal node requests a firmware update by broadcasting a version check message to other blockchain network nodes, which respond to the message. If the first response comes from a verification node, then the verification node checks whether the firmware of requesting normal node already is up-to-date. If the firmware is up-to-date, then the verification node checks integrity of the firmware. If the requesting normal node’s firmware is not up-to-date, the responding verification node downloads the latest firmware version to the requesting normal node. If the first response comes from another normal node, then the responding normal node compares the version of its firmware with the requesting node’s firmware version. If the firmware versions are the same, then a lightweight Proof-of-Work mining procedure in blockchain network checks the correctness of the verifier of the requesting node’s firmware. Six confirmations from other blockchain network nodes prove the correctness of the verifier. If the firmware versions are different, then a verification node downloads an up-to-date firmware file to the normal blockchain node whose firmware version is older. The proposed firmware update scheme uses a blockchain block scheme, where each block has a header and a verification field. In the header is stored the size and version of the block, a hash of the header of the previous block, and the root of the Merkle hash tree in the verification field.

The CSA Blockchain/Distributed Ledger Technology Working Group published in 2018 a report “Using Blockchain Technology to Secure the Internet of Things” [23]. In a preferred communication model, each IoT device is a blockchain network node hosting the full ledger of transactions and is capable of participating in blockchain transaction validation and mining. Because of the limited processing, storage, and power resources of most IoT devices, the report proposes a communication model where IoT devices are clients with Application Programming Interfaces (APIs) to blockchain nodes in a cloud based blockchain network service. An IoT device sends digitally signed data from its API to a blockchain network node for processing. A trusted secure communication channel is required between the IoT device and the blockchain network node. The blockchain ledger can store the last version of validated IoT firmware or its hash. An IoT device requests its blockchain network node to deliver, from the transaction ledger, the latest firmware version or the hash of this version. If the blockchain network node delivers a hash, then the IoT device retrieves the latest firmware version from a cloud service and checks if the hash of the retrieved version matches the hash delivered by the blockchain network node.

B. Development of PoC for our Solution

The literature review on the security of IoT software updates shows that research on this topic area has yet to receive the focus it deserves. Although several expressed opinions exist, a universal method (de-facto standard) for solving the problem does not yet exist. That secure IoT device updates and management is problematic or even unsustainable has been established, still very few, if any, solutions exist for either the open source community or for commercial manufacturers to automate and secure software updates to IoT devices in a transparent fashion. The papers reviewed provide several good ideas for further study to identify single case mechanisms for providing IoT update services utilizing DLT-based version control. We proceed through an exploratory approach aimed at understanding the engineering demands of such systems by constructing a PoC backend as an initial step [5].

In our distributed IoT architecture proposal, shown in Fig. 2, IoT nodes transmit their log data to a distributed and replicated data store. The data store exists outside the limited nodes and utilizes a P2P protocol. Utilization of different data stores depend on requirements, such as scalability, speed, or post-processing. A suitable batch-based solution may be the IPFS or a proprietary P2P data transfer protocol. If a streaming solution is required, then the use of a decentralized data and analytics marketplace such as Streamr [24] is an option. Smart contracts executed on top of a DLT implementation may authorize IoT devices and furthermore offer device management, e.g., issue management commands. Implementation of an automatic service for IoT device firmware updates may be similar. Storing the latest version of a binary update file in IPFS and in a smart contract store an IPFS immutable content address that allows the node to query correct IPFS file and firmware signature to confirm file integrity. This tells the IoT node how to access IPFS files and how to perform verification of the needed update. A different solution is to make use of an IPNS hash that points to the latest IPFS hash. The third solution is to mix both approaches. As these systems require two different logins for a manufacturer to share an update, 2-step verification is achievable by using both techniques (smart contract and IPFS) in combination and then compare the content hashes to the downloaded file update hash.

For the future, we consider it important that a manufacturer may want to offer a service contract to any IoT system maintainer/owner. Currently, a significant problem is that IoT nodes have no long-term support as the manufacturer often fails to get financial compensation for updating firmware once the product enters a maintenance/archival phase. A smart contract providing the manufacturer with a decentralized platform for selling firmware updates could implement this business model. An automated update function and contract resolution can be provided to any IoT node maintainer, either on a node basis (number of nodes) or on a network basis (maintaining organization).

C. *Explored Mechanisms and Methods*

This section is devoted to reviewing the technical mechanisms used for the implementation of the frontend interface for the software update manufacturer and the backend. The frontend utilizes a DApp that allows the software developer to deploy new software releases to the platform. A DApp is a stateless web application stored on IPFS and is executable without any dedicated server. This is possible by creating a web application that is self-contained and run within a browsing session initiated by a user. Hence, no server-side processing is required as the client downloads and executes the application. Routines in a JavaScript API library [25] push data to and pull data from IPFS node storage.

The DApp can be, while it is running, as dynamic as a traditional web application; however, from the statelessness follows, that no collected data is normally sent back to a server and stored when the browser is shutdown. Naturally, in the future, there will be more advanced use cases as well, but the idea of decentralized platforms such as ours is to avoid centralized processing that introduces dependencies, bottlenecks and transparency concerns. User authentication occurs before publishing new updates through the IPFS node and through Ethereum [26]. In addition to maintaining the latest content update in IPNS we also propose to store it in a smart contract. The main reason for this is that dual verification can ensure either a two-factor authentication or that the development team can share the IPFS node key for administration purposes, while the final software update release will require the Ethereum key as well. The smart contract is also usable for auditing purposes, as each published directory hash (i.e., a combined hash taken of all files in a directory each time it changes) is stored on the blockchain. In Fig. 3, we present an information flow for the backend of the DApp for releasing software updates. The PoC proposal makes use of a smart contract for guaranteeing revision history and IPFS for file update distribution and file history record.

VI. CONCLUSIONS AND FUTURE DEVELOPMENT

An important task in keeping devices secure in IoT systems, is by ensuring automatic and secure delivery of updates. These challenges involve version control of the software intended to run on the edge nodes, confirming installed software and hardware versions, and linking the versioning data to usage data that may reveal patterns and storing the data that allows auditing of the system. Because IoT networks are distributed/decentralized (depending on network topology choices), we find that not only new technologies, but also new methods for securing IoT devices are needed.

Our paper presents an initial PoC and explores some of the important mechanisms involved in creating a method that offers a more transparent version control of updates than today’s services that are conceptually centralized. Our solution does not rely on a central node for distributing updates as IPFS handles file distribution and Ethereum smart contracts handle version management. Our continued development will focus on creating a fully secured and automated process for management of IoT software updates management and on verifying this process with IoT device integration.

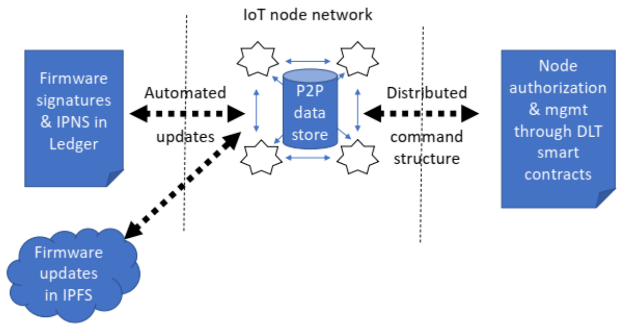
IoT node network
Firmware signatures & IPNS in Ledger
Automated updates
P2P data store
Distributed command structure
Node authorization & mgmt through DLT smart contracts
Firmware updates in IPFS

Figure 2. Proposed architecture solution, integrating IoT and DLT.

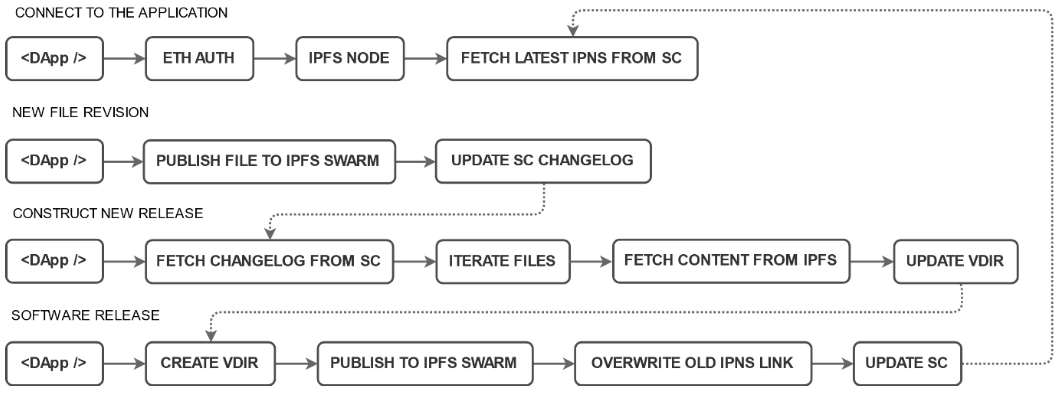

Figure 3. DApp information flow.

REFERENCES

[1] IBM Rational ClearCase. [Online]. Available from: https://www.ibm.com/fi-en/marketplace/rational-clearcase 2019.04.08

[2] Git–fast-version-control. [Online]. Available from: https://git-scm.com 2019.04.08

[3] Built for developers. [Online]. Available from: https://github.com/ 2019.04.08

[4] IPFS. *IPFS is the Distributed Web*. [Online]. Available from: https://ipfs.io/ 2019.04.08

[5] R. J. Wieringa, Design Science Methodology for Information Systems and Software Engineering. Berlin, Heidelberg: Springer-Verlag, 2014

[6] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, “Internet of things: Vision, applications and research challenges” Ad Hoc Networks, vol. 10, no. 7, pp. 1497-1516, 2012.

[7] O. Vermesan and P. Friess, Internet of Things - Global Technological and Societal Trends From Smart Environments and Spaces to Green ICT, Denmark: River Publishers, 2011.

[8] M. Westerlund, “A study of EU data protection regulation and appropriate security for digital services and platforms,” Doctoral Dissertation, Åbo Akademi University, Åbo, Finland, 2018.

[9] M. Giles. *For safety’s sake, we must slow innovation in internet-connected things*. [Online]. Available from: https://www.technologyreview.com/s/611948/for-safetys-sake-we-must-slow-innovation-in-internet-connected-things/ 2019.04.08

[10] J. Condliffe. *Should IoT Devices Come with Public Safety Warnings?* [Online]. Available from: https://www.technologyreview.com/the-download/609124/should-iot-devices-come-with-public-safety-warnings/ 2019.04.08

[11] M. Westerlund, M. Neovius, and G. Pulkkis, “Providing Tamper-Resistant Audit Trails with Distributed Ledger based Solutions for Forensics of IoT Systems using Cloud Resources.” International Journal on Advances in Security, vol.11, no. 3 and 4, pp. 288-300, 2018

[12] CSA cloud security alliance. [Online]. Available from: https://cloudsecurityalliance.org 2019.04.08

[13] CSA cloud security alliance. *Recommendations for IoT Firmware Update Processes*. [Online]. Available from: https://downloads.cloudsecurityalliance.org/assets/research/internet-of-things/recommendations-for-iot-firmware-update-processes.pdf 2019.04.08

[14] IETF. *Software Updates for Internet of Things (suit)*. [Online]. Available from: https://datatracker.ietf.org/wg/suit/about/ 2019.04.08

[15] B. Moran, M. Meriac, H. Tschofenig, and D. Brown. *A Firmware Update Architecture for Internet of Things Devices. draft-ietf-suit-architecture-02.* [Online]. Available from: https://datatracker.ietf.org/doc/draft-ietf-suit-architecture/ 2019.04.08

[16] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2008. [Online]. Available from: https://bitcoin.org/bitcoin.pdf 2019.04.08

[17] M. Westerlund and N. Kratzke, “Towards Distributed Clouds,” Proc. 16th International Conference on High Performance Computing & Simulation (HPCS), IEEE Press, July 2018, pp. 655-663, doi:10.1109/HPCS.2018.00108.

[18] Data done differently. [Online]. Available from: https://www.streamr.com/ 2019.04.08

[19] J. Ramos. *Blockchain: Under the Hood*. [Online]. Available from: https://www.thoughtworks.com/insights/blog/blockchain-under-hood 2019.04.08

[20] E. Feig, “A Framework for Blockchain-Based Applications,” arXiv:1803.00892 [cs.CY], 2018. [Online]. Available from: https://arxiv.org/abs/1803.00892 2019.04.08

[21] K. Christidis and M. Devetsikiotis, “Blockchains and smart contracts for the Internet of Things,” IEEE Access, vol 4, pp. 2292–2303, 2016, doi: 10.1109/ACCESS.2016.2566339.

[22] B. Lee and J.-H. Lee, “Blockchain-based secure firmware update for embedded devices in an Internet of Things environment,” The Journal of Supercomputing, pp. 1–6, 2016, doi: 10.1007/s11227-016-1870-0.

[23] CSA cloud security alliance. *Using Blockchain Technology to Secure the Internet of Things*. [Online]. Available from: https://downloads.cloudsecurityalliance.org/assets/research/blockchain/Using_BlockChain_Technology_to_Secure_the_Internet_of_Things.pdf 2019.04.08

[24] IPFS Documentation - IPNS [Online]. Available from: https://docs.ipfs.io/guides/concepts/ipns/ 2019.04.08

[25] A client library for the IPFS HTTP API, implemented in JavaScript. [Online]. Available from: https://github.com/ipfs/js-ipfs-http-client 2019.04.08

[26] Ethereum JavaScript API. [Online]. Available from: https://github.com/ethereum/web3.js/ 2019.04.08