Resilient, Auditable and Secure IoT-Enabled Smart Inverter Firmware Amendments With Blockchain

Akkaoui, Raifa; Stefanov, Alexandru; Palensky, Peter; Epema, Dick H.J.

DOI
10.1109/JIOT.2023.3321954

Publication date
2023

Document Version
Final published version

Published in
IEEE Internet of Things Journal

Resilient, Auditable, and Secure IoT-Enabled Smart Inverter Firmware Amendments With Blockchain

Raifa Akkaoui, Alexandru Stefanov, *Member, IEEE*, Peter Palensky, *Senior Member, IEEE*, and Dick H. J. Epema

***Abstract*—The solar industry in residential areas has been witnessing an astonishing growth worldwide. At the heart of this transformation, affecting the edge of the electricity grid, reside smart inverters (SIs). These IoT-enabled devices aim to introduce a certain degree of intelligence to conventional inverters by integrating various grid support capabilities (e.g., voltage and frequency control). However, with the remarkable automation of these devices come enormous security risks. Thus, rising rates of vulnerabilities have increased the necessity for designing resilient, auditable, and secure SIs' firmware over the air (FOTA) amendment schemes suitable for this heterogeneous SIs-based ecosystem. In this regard, we propose leveraging blockchain as an innovative technology to guarantee these cybersecurity requirements. In this article, we present the design of a distributed FOTA scheme, namely, RASSIFAB, governing the process of amending SIs' firmware within residential areas in an immutable and scalable manner. The scheme was implemented on a blockchain test network to assess its functionalities and performance. We also carried out a security evaluation to determine whether RASSIFAB is resistant to various identified threats. The obtained results confirm that the scheme is efficient and sound. They also indicate that RASSIFAB ensures reliable and authentic firmware amendments even with malicious insiders, differentiating our framework from the existing ones.**

***Index Terms*—Blockchain, distributed energy resources (DERs), firmware, Internet of Things (IoT), photovoltaic, security, smart inverter (SI).**

I. Introduction

PHOTOVOLTAICS are estimated to account for nearly 60% of the additional capacity of renewable energy resources worldwide by 2025. In particular, according to the *Renewables* report published by the International Energy Agency, the global share of rooftop solar panels installed within residential areas rounded up to 30% of the total solar panels' deployment in 2020 [1]. This eventually led to the advancement and adoption of innovative technologies to tackle the effect of the proliferation of these distributed energy resources (DERs) and smooth their integration within the low-voltage (LV) networks of the electricity grid. At the heart of these ingenious ever-evolving technologies, we find solar smart inverters (SIs). These Internet of Things (IoT)-enabled devices stretch far beyond their basic functionalities as they can additionally provide ancillary grid services (e.g., voltage/frequency support and ride-through capabilities), which are recently becoming mandated in the revised versions of existing DERs technical standards as well as several grid codes (e.g., IEEE 1547-2018 [2] and the German grid code VDE-AR-N 4105 [3]).

A. Motivation

It is undeniable that the Internet-of-Energy paradigm steered the enhancement of DERs' integration through the automation of the power grid's infrastructure and the implementation of DERs management systems (DERMSs) that enable a wide range of capabilities through (near) real-time communication. Nonetheless, its lack of cybersecurity inherited from the IoT paradigm has raised some eyebrows within network operators, i.e., transmission system operators (TSOs) and distribution system operators (DSOs). In fact, these IoT-enabled DERs could potentially be leveraged using their vulnerabilities (e.g., Mirai botnet [4], [5], BlackIoT [6]) as an attack-vector to disrupt the power supply by causing voltage issues and frequency instabilities, which could lead to power outages.

On the one hand, the vulnerabilities of IoT devices and SIs in particular, have been extensively documented in [7] and [8], such as spoofing attacks where the hacker would masquerade as the DERs' aggregator (DERA) and alter critical settings of the SIs (e.g., Volt/Var or Volt/Watt curves). In addition, Tertytchny et al. [9] demonstrated the possibility of launching a man-in-the-middle (MitM) attack on commercial SIs supporting ancillary services, which could lead to an intentional false-tripping of whole feeders, eventually causing a regional blackout. Meanwhile, firmware over-the-air (FOTA) security updates and/or patches are amongst the critical procedures within cyber–physical systems. For instance, the 2015 cyberattack on the Ukrainian power grid was further complicated when the attackers were able to alter (or reverse engineer) the firmware of the serial-to-Ethernet converters within dozens of substations, which prevented the operators from implementing any remote procedure for recovery [10]. Furthermore, Konstantinou and Maniatakos [11] studied the impact of reverse engineering the firmware of power grid devices on an IEEE 14-bus test system. Specifically, on commercial protection relays within substations to alter the normal behavior of circuit breakers, which resulted in cascading failures. In addition, CVE-2017-9860 [12] is a vulnerability discovered that

Manuscript received 31 October 2022; revised 12 April 2023; accepted 28 September 2023. Date of publication 3 October 2023; date of current version 21 February 2024. *(Corresponding author: Raifa Akkaoui.)*

Raifa Akkaoui, Alexandru Stefanov, and Peter Palensky are with the Department of Electrical Sustainable Energy, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: R.Akkaoui@tudelft.nl; A.I.Stefanov@tudelft.nl; P.Palensky@tudelft.nl).

Dick H. J. Epema is with the Department of Software Technology, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: D.H.J.Epema@tudelft.nl).

Digital Object Identifier 10.1109/JIOT.2023.3321954

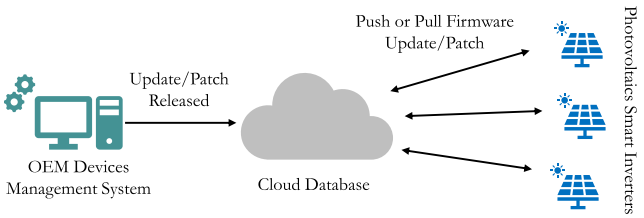

Fig. 1. Conventional FOTA amendments for SIs.

could potentially allow an attacker to update an SI's firmware without proper authentication.

On the other hand, FOTA amendments conventionally rely on a cloud database from where the binary files are downloaded, as illustrated in Fig. 1 for the case of residential SIs. This storage could be either on-premises meaning it is hosted and administered internally by the manufacturers themselves using their on-site information technology resources, or offloaded to a third-party cloud service provider. The advantage of the first approach is that critical and proprietary data regarding the firmware's code are protected by the organization following its internal guidelines; however, the vendor could still be the target of a cyberattack either from an external or insider adversary that may tamper with the stored files by injecting some malware. Meanwhile, the second approach gives many economical and technical advantages, such as low cost in terms of maintenance and high redundancy that guarantees availability with an uptime of nearly 99.99%; nonetheless, the approach raises a series of concerns, such as privilege escalation attacks and privacy due to security breaches [13]. Therefore, rather than focusing on the single point-of-failure of cloud-based systems in the sense of having the database going down. It is rather essential to focus on the over-reliance on third-parties, or broadly speaking on a single governing entity (which could also be the manufacturer itself in this case), this may create a single point-of-weakness leading to an inherent bias and selective disclosure [14].

Therefore, a consortium blockchain-based framework where various entities all share the governance of the whole ecosystem and no participant alone is in charge of the decision making seems to be beneficial and a perfect fit. Besides, the blockchain ledger would also serve as an immutable audit trail for all data and actions undertaken regarding any issued firmware update, that could be used for future digital forensics analysis. Whereas the interplanetary file system (IPFS) would serve as a storage layer for the blockchain focusing on the reliability and immutability of the firmware files [15].

B. State-of-the-Art

Securing firmware amendments has attracted tremendous attention from the research community. Some approached the security challenges of the procedure through hardware-assisted schemes, such as trusted execution environments or physical unclonable functions (PUFs) [16], [17], [18], whereas others introduced variations of cryptographic mechanisms, such as checksums and code signatures for authenticity [19], [20], [21] to secure the conventional centralized way of FOTA amendments. Nonetheless, these schemes still rely on individual-based decision making due to their centralized architecture.

Blockchain, hailed as one of the utmost disruptive technologies, was harnessed for various IoT-based applications [22], [23], but in particular to tackle the limitations of conventional centralized-based FOTA schemes. For instance, Pillai et al. [24] proposed an Ethereum-based scheme for updating IoT devices' firmware. Where upon the release of a new firmware the vendor node part of the blockchain creates and deploys a smart contract (SC) to record the data on the shared ledger. However, the scheme is based on the assumption that each original equipment manufacturer (OEM) has a single node responsible for storing the firmware data on the blockchain, thus if this single node is hacked the devices might download binary files that have been tampered with. Furthermore, in [25], an Ethereum-based framework to manage FOTA updates for medical IoT devices was proposed, where each OEM node would send a transaction to the blockchain network containing the URI of the firmware to the gateway of the devices. Nonetheless, the scheme still relies on the centralized repositories of the OEMs, which could eventually be a target of a cyberattack affecting the reliability of critical patches. Meanwhile, Bere et al. [26] focused on the firmware check and recovery procedure for SIs using blockchain. The distributed ledger is used to store in an immutable way the metadata of the firmware. Whereas the onboard security module connected to the device is responsible for periodically checking if the current version of the installed firmware matches the one recorded by the vendor. Still, the security of the scheme relies on the data written on the ledger, which is recorded by a single vendor client. In addition, the framework is based on a private blockchain managed by a single OEM, meaning that the proposed solution is more suitable for homogenous ecosystems. However, residential LV networks equipped with a large share of DERs are characterized by devices coming from a myriad of vendors. Thus, the heterogeneity aspect of these devices should also be accounted for while designing an FOTA framework.

C. Contributions

In this article, we aim to address the aforementioned gaps by proposing a resilient, auditable, and secure SIs firmware amendments with blockchain (RASSIFAB) framework suitable for heterogeneous ecosystems. The framework is based on two developed SCs; the first is used for the initialization phase including the registration of the entities part of the system (i.e., TSOs, DSOs, and OEMs) as well as the IoT devices (i.e., SIs). Whereas the second is used to monitor the status of the SIs' firmware and issue new updates or patches with a distributed and autonomous checksum verification on-chain of the firmware metadata. In addition, to address the OEMs' over-reliance on single trusted third parties, we propose to utilize a P2P file-sharing system based on the IPFS, where different OEMs form a coalition to store the encrypted binary firmware files. To the best of our knowledge, there exists no blockchain-based scheme dedicated for SIs' FOTA amendments that considers the heterogeneity aspect of these

devices within the edge of the SG as well as the threat of insider attacks posed by the various OEMs part of the ecosystem. To summarize, the major contributions of this article are as follows.

1) We propose a distributed access control-based FOTA amendment scheme for heterogeneous SIs within the residential LV networks of the SG that addresses the threat of insider attacks.
2) We implement a proof-of-concept of RASSIFAB to demonstrate its feasibility, with a performance evaluation of the framework based on various metrics as well as a security analysis following the defined threat model in order to assess its efficiency and soundness.

D. Paper Organization

The remainder of this article is organized as follows. Section II represents some security standards and best practices for firmware amendments in general and DERs in particular. In Section III, we present the architecture design of our proposed blockchain-based system, its threat model, and goals. Section IV details the system's workflow and functionalities. The implementation of RASSIFAB, including the performance as well as security analysis, is discussed in Section V. The related work is examined in Section VI with an in-depth comparison between our scheme and the existing ones as well as the limitations of RASSIFAB. Finally, Section VII concludes this article with some future directions.

II. Security Standards and Best Practices

Securing FOTA for IoT devices in general and the cybersecurity aspect of DERs have been the focus of various standardization initiatives by multiple international agencies, including the Internet engineering task force (IETF), the National Institute of Standards and Technology (NIST), the International Electrotechnical Commission (IEC), the Institute of Electrical and Electronics Engineers (IEEE), the Underwriters Laboratories (UL), and so forth. Where the major goal of these initiatives has been to provide a shared platform for debate, with a set of explicit best practices and recommendations that would further enhance the security of IoT devices in general and DERs in this particular use case.

For instance, the IETF RFC 8240 [27] is a summary of the IoT software update workshop held in 2016 and organized by the Internet architecture board. The workshop aimed to discuss several challenges and vulnerabilities relevant to the procedure of updating and/or patching IoT devices' firmware. This eventually led to establishing the IETF software updates for IoT (SUIT) working group [28], which focuses on defining various components of a security update solution (e.g., transport protocols, firmware's metadata, cryptographic mechanisms for integrity checks, etc.). Among the tasks completed by the working group is the IETF RFC 9019 [29], which defines a secure architecture for IoT devices' firmware. In addition, the IETF RFC 4108 [30] introduces a format for firmware packages' digital signatures utilizing cryptographic message syntax, that contains an identifier, the firmware's version, a description of the package, certificate of the signer, etc. Nonetheless, the approach is based on a single trust anchor consisting of a public-key signing mechanism to validate the hash of the firmware as well as the identity of the organization that signed the package using its private key. Meanwhile, patch management for industrial automation and control systems was also addressed in the IEC 62443 Series [31].

Besides, code signing aims to ensure the firmware's integrity and nonrepudiation. It is based on digital signatures for the verification of provenance of the data (i.e., OEMs) and checksum mechanisms (e.g., MD5, SHA-1, SHA-256, etc.) to validate that the data was not subject to any modification during the transit phase. As presented in the NIST white paper [21], the procedure relies on three main entities, i.e., developer, signer and verifier. The developer is responsible for creating the firmware code, ensuring it is bug-free using the OEMs auditing tools and then sending it to the signer. The latter requires the authentication of the developer before signing the data using the private key of the OEMs tied to a certificate authority. However, the procedure could be vulnerable to various threats. For instance, it is likely that the signed version of the firmware contains a malicious code embedded in it through an insider attack, the theft or leakage of private signing keys, which will result in compromising the software supply chain of the OEM in a similar manner to what happened during the notorious SolarWinds attack [32].

As for DERs' standards, the IEEE 1547 [2] recognizes that cybersecurity is a critical concern while considering DERs connected to larger monitoring and control systems associated to the electricity grid, without directly addressing this challenge. Nonetheless, the working group is currently drafting a guide for DERs cybersecurity, i.e., IEEE P1547.3 [33], which touches on the requirements for patching their IoT-enabled devices. Meanwhile, the UL 1741 standard [34] addresses the requirements for inverters as part of the interconnection systems used with DERs; however, with no explicit details regarding the security aspect. Nevertheless, the UL recently announced that they are working jointly with the U.S. Department of Energy's National Renewable Energy Laboratory on a cybersecurity certification standard for DERs that also addresses firmware updates of SIs [35], which again demonstrates the criticality of this matter.

Thus, by taking into account the discussed standards and their relevant requirements in terms of DERs' firmware security (in particular residential SIs), we design our framework not only in a way that incorporates the identified best practices and recommendations derived but also addresses some of the limitations of current practices. Fig. 2 represents a high-level representation of our framework, which can be abstracted into two major components: 1) the blockchain network and 2) the distributed shared storage. Furthermore, the blockchain network can be divided into two sublayers as it is a combination of lightweight nodes (i.e., SIs) and full nodes (i.e., TSOs, DSOs and OEMs). These nodes all have different access rules and privileges, which are defined in the SCs developed. We note that the in-depth details of the proposed architecture model and SCs would be further discussed in Sections III and IV.

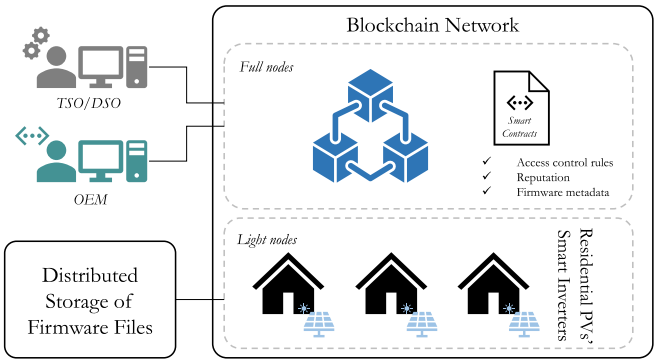

Fig. 2. RASSIFAB abstracted system model.

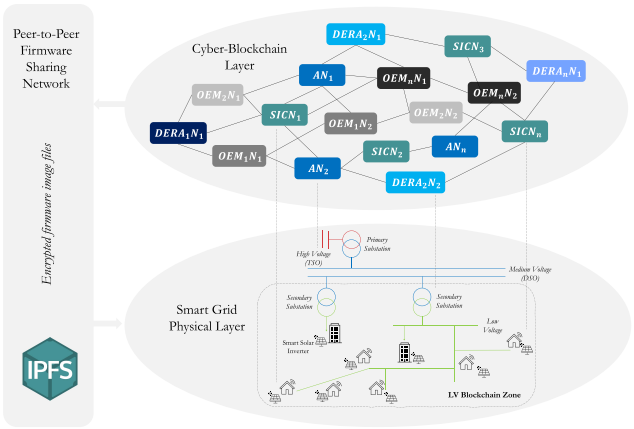

Fig. 3. Detailed architecture design of RASSIFAB.

III. Problem Formalization

In this section, we present the system model, the plausible threats, as well as the design goals of RASSIFAB.

A. System Model

The architecture design of our proposed FOTA framework is illustrated in Fig. 3, which can be divided into three main different components. Namely: 1) the SG physical layer which represents the actual edge devices of the LV network (in this case SIs); 2) the cyber-blockchain layer encompassing the various entities part of the P2P network as well as the SCs defining all data structures, metadata, functions, and modifiers detailing the access control rules and policies of RASSIFAB,; and 3) the distributed consortium firmware file-sharing system. Hereafter, we introduce the various stakeholders part of the framework and detail their responsibilities.

1) Transmission System Operators: In the context of the electrical grid, a TSO is an entity delegated with the task of transmitting electrical power from generation plants to the distribution networks in various regions. In our framework, the TSOs being the highest authority in a power grid operational system, are delegated with the administrative tasks and workflows of setting up the blockchain-based ecosystem. They are responsible for managing the administrator nodes (ANs) part of the blockchain framework, which are in charge of the first deployment of the SCs to the consortium blockchain network as well as the execution of certain functions during the initialization phase of the system (which will be detailed in Section IV-A). Besides, the framework is based on the idea of splitting the LV network of the power grid into several semiindependent blockchain zones (BZs) geographically distributed. Thus, we assume that each zone would fall into the control of one single TSO, which will be responsible for managing a set of ANs.

2) Distribution System Operators: The DSOs are the entities operating the distribution networks, whose role was limited to supply electricity to end consumers. However, with the proliferation of DERs and the introduction of a bi-directional flow of energy. DSOs are shifting from their traditional tasks to a more smart and intelligent control, by deploying DERA nodes (DERANs) with DERMSs to control these renewable resources dispatched across the LV networks. The DSOs first relied on smart meters for data collection, however, with the introduction of SIs those are also being leveraged to monitor the state of the LV network and coordinate the multiple DERs at the edge of the SG. In addition, these SIs within residential areas are connected to the grid and support ancillary services. Hence, maintaining an immutable inventory list of these intelligent devices as well as keeping track of their firmware updates is of tremendous criticality from the viewpoint of a DSO. This would ensure the security and reliability of the measurements collected and also protect these devices from being hacked or compromised, e.g., firmware downgrade attacks or reverse engineering that would alter their normal operation. In our proposed scheme, each BZ, depending on its location and size (e.g., neighborhood, district, city, etc.), would be supervised by either a single or multiple DSOs responsible for managing their respective DERANs in the blockchain framework.

3) Original Equipment Manufacturers: Typically, the LV network within residential areas is characterized by its heterogeneity in regard to the OEMs from which the SIs installed are originating. In fact, it is no surprise that each household would invest in a different inverter based on costs or personal preferences and those would widely differ from one prosumer to another. Meanwhile, each OEM is responsible for issuing firmware updates and/or patches to its proprietary devices (i.e., SIs in this particular case) to guarantee their normal and correct behavior as well as cybersecurity. In our scheme, and by taking into account the heterogeneity aspect of the SIs within this ecosystem, each BZ would encompass a set of various OEMs labeled as *OEM* in which each manufacturer is denoted as OEM_i with $i \in \{1, \ldots, |OEM|\}$. Whereas, each OEM_i is responsible for a set of blockchain nodes labeled as OEM_iN in which each manufacturer node is denoted as OEM_iN_j with $j \in \{1, \ldots, |OEM_iN|\}$. In addition, our scheme is also leveraging a distributed consortium file-sharing system based on the IPFS. Where for each OEM, a number of blockchain nodes are also designated as the IPFS storage nodes.

4) Prosumers: In our framework, each residential prosumer is supposed to have an SI connected to its household's solar panel. As SIs are characterized by limited computation and storage capacities, they are part of the blockchain framework as light nodes in contrast to the aforementioned nodes.

Basically, they neither store a full copy of the distributed blockchain ledger nor participate in the consensus to create blocks. However, they keep track of the headers of each block which contain the Merkle roots of a given transaction pool. Thus, allowing them to verify the credibility of a transaction by mounting its hash until reaching the root of the Merkle tree of the block containing that specific transaction. In addition, the SIs client nodes (SICNs) are also running an IPFS client node that allows them to download the firmware files from the distributed storage system.

Meanwhile, it is worth stressing that configuring a blockchain node using the existing SIs' hardware or software might be challenging due to some dependencies' issues. Thus, it is possible to have those SIs linked, using an Ethernet connection, to a designed system on-chip blockchain unit built using a Raspberry Pi for instance, that would host the blockchain-based capabilities defined in our scheme as well as the IPFS client. However, we should specify that the technical details and implementation of this fall beyond the scope of this article and could be the focus of future work.

B. Threat Model

The aim of this section is to identify the threats from which FOTA updates in the case of SIs are vulnerable to and eventually derive the security requirements and goals that must be satisfied. The threat model upon which RASSIFAB is built is following the S.T.R.I.D.E. approach [36], which divides security risk into six main classes ($\mathcal{C}$), i.e., Spoofing of identity (targeting authentication, denoted later as $\mathcal{C}1$), Tampering with data (against integrity, denoted later as $\mathcal{C}2$), Repudiation (affecting accountability, denoted later as $\mathcal{C}3$), Information leakage (disturbing confidentiality, denoted later as $\mathcal{C}4$), Denial of service (influencing availability, denoted later as $\mathcal{C}5$), and finally Escalation of privilege (hindering authorization, denoted later as $\mathcal{C}6$). In what follows, we detail the various security threats and attacks that could be exploited during the procedure of amending SIs' firmware and we also indicate under which class of the S.T.R.I.D.E. model they fall.

1) Firmware Downgrade Attack [$\mathcal{C}5, \mathcal{C}6$]*:* In this scenario, an attacker tries to push an old, but correct firmware to the SI with a valid signature from the OEM. Suppose that particular version of the firmware is known to contain a vulnerability; the attacker can then try exploiting it by opening a backdoor that may eventually allow him to gain full control over the SI with the right escalation of privilege. Therefore, the attacker can disconnect all inverters that have been a target of this attack, causing a Denial-of-Service (DoS) attack.

2) Firmware Mismatch Attack [$\mathcal{C}5$]*:* In this attack, the malicious user tries to send a valid firmware but for a different device; for instance it could be a firmware of a smart meter sent to an SI from the same vendor. If the device does not require a rigorous check of the firmware file, then the update is most likely to be accepted as its code signature is valid. The impact of such an attack on SIs can scale from minor misbehavior or abnormal functioning of the inverters to rendering the devices totally inoperable. In fact, this type of attack has been reported recently, due to a human error, where an employee entered the wrong identifier of the devices to be updated, leading to a mismatched firmware being sent to multiple microwaves causing them to crash and stop functioning [37]. Thus, it is also important to incorporate the risk of human error in our proposed scheme and mandate the verification of the FOTA updates by various authorized authors to minimize the likelihood of this risk.

3) MitM or Redirection Attack [$\mathcal{C}2, \mathcal{C}5$]*:* If the transportation phase of the FOTA update is not properly configured and secured, an attacker would be able to launch an MitM attack by redirecting the SIs to download a malicious firmware file from a corrupted server.

4) Firmware Reverse Engineering [$\mathcal{C}2, \mathcal{C}4$–$\mathcal{C}6$]*:* If the firmware files are not properly encrypted, a malicious user can easily have access to them. To then reverse engineer those files, introducing new vulnerabilities or backdoors to fully gain remote control over the SIs capabilities. Thus, the confidentiality of the firmware binary files should be guaranteed.

5) Supply Chain Attack [$\mathcal{C}1$–$\mathcal{C}6$]*:* In this scenario, the attacker is able to infiltrate the OEM's system and deploy a malicious malware that would alter the firmware before sending it to the end users. Besides, the attacker could also hijack the account of one of the authors responsible for signing the firmware package and use the leaked key to dispatch the compromised firmware to the SIs [32].

C. Design Goals

Based on the aforementioned threats identified, this section presents the security goals ($\mathcal{G}$) and requirements that RASSIFAB must satisfy.

1) Authentication, Authorization, and Accountability [$\mathcal{G}1$]*:* These refer to a set of techniques utilized to regulate and track access control within a system. In our proposed scheme, first, authentication refers to the ability of authenticating both the OEMs, sending the firmware with its metadata, as well as the SIs devices. Second, authorization signifies that only authorized entities can execute certain functionalities based on their privilege and access rules defined. For instance, an OEM B should not be allowed to send an update to devices manufactured by OEM A. Last, transparency and accountability are critical features that must be ensured in this scenario (i.e., FOTA), as each update or patch sent must be recorded in an immutable manner and verified by other entities part of the framework based on their roles and affiliations. If we consider the case of political cyberattacks, an OEM C of a given country could try to update specific devices with a corrupted firmware. Thus, if the scheme is based on a consortium blockchain involving various OEMs and other stakeholders (i.e., TSOs and DSOs), the OEM C would be unable to alter the shared copy of the ledger to hide all traces of the attack, which would guarantee accountability.

2) Firmware Integrity and Nonrepudiation [$\mathcal{G}2$]*:* The reliability of an FOTA update has to be provable, meaning that the update must be signed with a digital signature (or preferably multiple signatures to increase the resilience of the procedure).

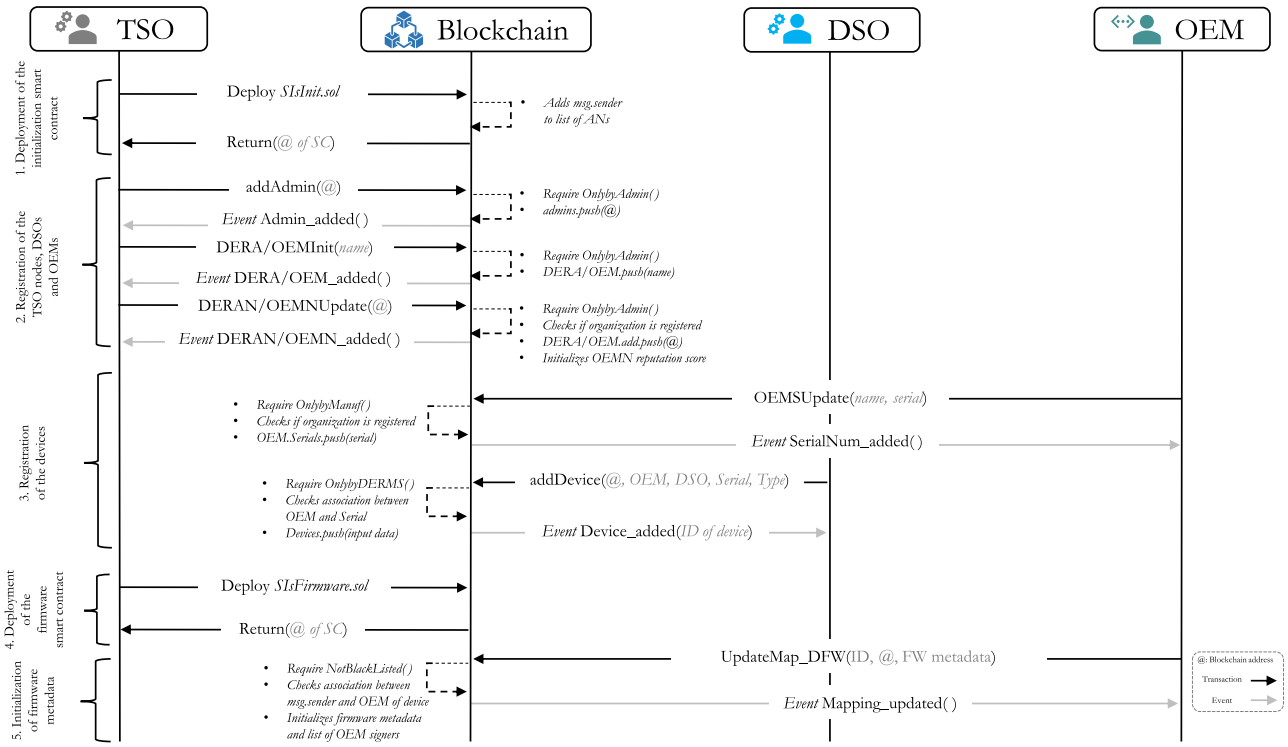

Fig. 4. Initialization and registration phase workflow of RASSIFAB.

Meanwhile, as the metadata of the update includes some relevant information about the firmware, it should also be authenticated. Thus, to minimize the effort of validating the signatures of both the metadata and firmware itself, we opted to tie both and include a digest or fingerprint (i.e., hash) of the binary image into the firmware metadata recorded on-chain that would eventually be signed by the OEM's authors.

3) Firmware Confidentiality [$\mathcal{G}3$]*:* The binary image of the firmware is required to be encrypted not only in order to protect the proprietary content of the file but also to stop attackers from reading it and eventually reverse engineer it to introduce some vulnerabilities.

4) Lightweight Security and Scalability [$\mathcal{G}4$]*:* As SIs are characterized by low computation as well as storage resources, the proposed scheme should incorporate lightweight mechanisms at the edge of the LV network. In addition, leveraging blockchain would inevitably introduce some scalability issues both in terms of transactions per second rate as well as the ledger's overhead. Thus, it is important to design the architecture of the FOTA framework in a way that would alleviate this burden and not over-stress the SIs, while at the same time guaranteeing the level of security needed within this heterogeneous ecosystem.

IV. Proposed RASSIFAB Scheme

In this section, we discuss the workflow of our proposed scheme by first detailing both the registration and initialization phases to set up the blockchain system. We then present the access control policies defined in the SCs and explain how the firmware update procedure for SIs is performed.

A. Initialization and Registration

Before joining the consortium blockchain-based framework, all participants are assumed to have generated locally their respective blockchain keys following the elliptic curve digital signature algorithm (ECDSA). These pairs of keys would be used to sign and authenticate transactions sent to the blockchain network as well as ensure the validation of all blocks generated. Then, the blockchain addresses would be derived from these public keys, which would serve as unique identifiers for each of the node parts of the framework in the implementation of the distributed access control. Meanwhile, the ANs, DERANs, and OEMNs are considered to be under the supervision of known organizations, hence, their public keys can be distributed and verified by means of a certificate authority for instance. For SIs, their public keys and addresses generated by the prosumers would be communicated to the DSOs and OEMs through secure channels for privacy and then used during the initialization phase.

The initialization steps of our proposed blockchain-based FOTA framework are illustrated in Fig. 4, which are divided into five main subphases. The first subphase involves the deployment of the *SIsInit.sol*[1] SC (which defines the structures used to map all stakeholders, SI devices, their metadata, and access rules) to the blockchain network by one of the TSO nodes (i.e., ANs) controlling a given BZ. If successful, the defined list of ANs in the SC would be automatically instantiated with the address of the node that deployed the contract (i.e., *msg.sender*). In addition, this node would receive the

[1]RASSIFAB source code. [Online]. Available at: https://github.com/Raakk/RASSIFAB/.

deployment address of the SC that would be eventually shared with the rest of the participants and used to initiate all interactions with the functions defined within the contract as well as access its associated storage.

The second subphase involves adding the various node parts of the blockchain framework. In fact, for the sake of cybersecurity and resilience, the contract enables updating the list of ANs, so that in case one is offline or faulty, others can still operate the system and ensure its continuity. Meanwhile, the DERAs and OEMs are also registered to the framework using *DERAInit()* and *OEMInit()* (which are functions that take as input the names of each entity, converted to *bytes8* type for optimization purposes, serving as unique identifiers for the implementation of the access rules). Whereas *DERANUpdate()* and *OEMUpdate()* are functions used to register the blockchain addresses of both the DERA and OEM nodes, respectively, while mapping them to their respective organizations. It is worth noting that these functions can only be executed by an AN leveraging the *onlyByAdmin()* modifier defined in *SIsInit.sol*.

The third subphase is focused on adding the multiple SIs to the consortium blockchain system. It starts by updating the list of the devices' serial numbers by each of the OEMs using the *OEMSUpdate()* function. Those serials would be later used to bind the devices with their respective OEM using the *addDevice()* function which returns a unique *Id* of the SI. The latter function is used to register the devices on-chain by inputting their metadata, i.e., blockchain address, the OEM who fabricated the device, the DSO managing the grid zone to which the SI is connected, the unique serial number of the device and its type. We should note that we included a variable to register the type of the device in order to guarantee the future extensibility of the scheme proposed here to encompass other devices, such as smart meters, sensors, etc. Both functions mentioned are restricted using the *verifyOEM()* and *onlyByDERMS()* modifiers, respectively.

Then, the fourth subphase starts by deploying the *SIsFirmware.sol* contract (that inherits the first one), which is used for issuing firmware updates and/or patches requests to the SIs. Similarly, the address of the SC is also broadcasted among all participants after its deployment. Then, the last subphase is mapping the already added devices in the previous contract with their current firmware metadata (i.e., version and hash of the installed firmware as well as the IPFS link from where the file could be recovered if needed) using the *UpdateMapp_DFW()* function. It should be noted that this function can only be executed by one of the OEM nodes of the SI by checking the association between the *msg.sender* and the OEM of the device using the *checkManuf()* modifier.

Meanwhile, each OEM_iN_j has a reputation score denoted as ρ encoded within the *SIsInit*.sol SC, with a value that ranges from 0 to 5. In fact, during the initialization phase performed by the ANs, each OEM_iN_j is assigned with an initial reputation score denoted as ρ_0 above the trust threshold of the system denoted as ρ_T, meaning that $\rho_0 > \rho_T$, as these nodes are assumed to be semitrusted. Then, based on the historical behavior of these manufacturer nodes within the framework, their reputation can be deducted if they misbehaved at any given round, where $OEM_iN_j \cdot \rho_{\text{time}} = OEM_iN_j.\rho_{\text{time}-1} - \delta$, with δ being fixed at 1. Specifically, misbehavior in our scheme is defined as sending a firmware update or patch transaction containing compromised metadata (i.e., wrong hash of the binary file or associated IPFS path) which will be detailed in the following section. Moreover, this reputation is used to revoke the nodes' access to the framework (in terms of sending transactions) once their score falls below ρ_T.

B. Firmware Update Procedure

The process of sending a firmware update to an SI belonging to OEM_i for instance, is depicted in Fig. 5, which is encoded in the *sendFWupdate()* function of the *SIsFirmware.sol* SC. Before the transaction gets to trigger the execution of the actual body of this function, certain requirements need to be fulfilled (which are defined within the contract's modifiers). First, the node sending this transaction, being $OEM_{i'}Nj$, should not be blacklisted (i.e., $OEM_{i'}Nj \cdot \rho > \rho_T$) and that the *msg.sender* address needs to satisfy the following condition $OEM_{i'}Nj \in OEM_iN$ i.e., $i' == i$. If any of these conditions is not satisfied the transaction would fail with a message error depending on which requirement was not met. Then, as this is an update rather than a patch, the version of the firmware must be higher than the one recorded on-chain (i.e., the current version installed on the device) for the transaction to be valid. The next step of the procedure is to check the SI's signers list, which is used to guarantee the BFT nature of the procedure.

If the signers' list is empty, then the temporary fields for the firmware metadata are initialized and $OEM_{i'}Nj$ is pushed into the list. Meanwhile, in case the list is not empty, the function checks whether the provided metadata are equal to the registered ones. If there is a mismatch between any of the new input data, $OEM_{i'}Nj.\rho$ would be deducted by δ and if the reputation is below ρ_T the node is automatically added to the blacklist and the transaction would fail. However, if the provided metadata are correct, the function checks if the majority of the nodes belonging to the SI's OEM have endorsed the same firmware metadata. Only then an event (i.e., *New_Firmware_Update()*) would be triggered automatically containing the new firmware metadata and the address of the device that requires an update. Meanwhile, if the metadata was not validated by the majority, the device would wait for the other OEM nodes. We note that the device's address is defined as an indexed parameter, thus allowing to leverage event listeners within scripts that would use this value. For instance, each SICN would listen to this event and get a notification only if the address mentioned in the event matches its own unique blockchain address.

After receiving the firmware metadata from the blockchain (i.e., hash of the firmware, IPFS link, encrypted blockchain secret key denoted as $S_{\text{key}}^{\text{Chain}}$ and version). The SICN starts by decrypting $S_{\text{key}}^{\text{Chain}}$ using its own *AES-256* preset key (e.g., embedded during the manufacturing process of the device). The obtained key is used to decrypt the firmware file downloaded from the IPFS link received, then the hash of the file

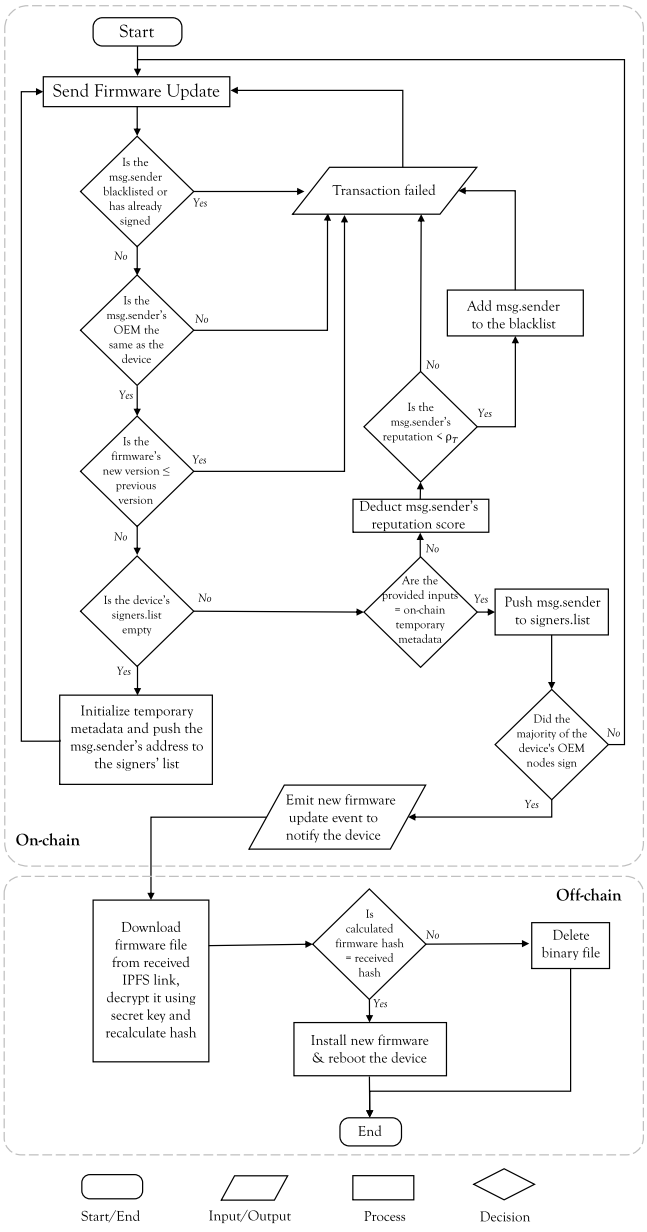

Fig. 5. Flowchart of the proposed blockchain-based FOTA update scheme.

is calculated. If the obtained hash is not the same as the one received from the blockchain, the file is then deleted; otherwise, the new binary file would be installed and the SI would reboot. We should note that we used this approach as the binary files are supposed to be encrypted before being pinned to the IPFS by the OEMs before starting an update request on the blockchain framework. Suppose that for each device, its firmware was encrypted using a unique key this would result in a storage overhead in terms of duplicated files. In fact, devices belonging to the same OEM and which are also the same model would have the exact same firmware. Thus, the file is encrypted with a single key (for the same set of devices) and this key is in turn encrypted using each of those devices' unique keys. The cyphered key is then sent through the blockchain as part of the firmware metadata.

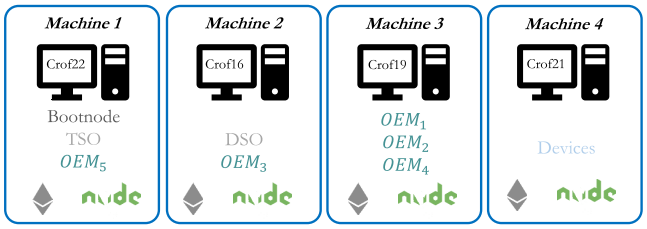

Fig. 6. Proposed RASSIFAB implementation using Ethereum.

V. Implementation and Analysis

Our proposed framework was implemented on a blockchain test network using Geth, being the Go implementation of Ethereum. The network was deployed on four Ubuntu virtual machines (located on different physical hosts) each with four vCPUs and 16 GB of RAM. The machines were then partitioned into multiple virtual Ethereum nodes with several accounts to emulate all entities of the system as illustrated in Fig. 6. Besides, the consensus we have opted to leverage is *Clique* which is a variation of PoA. The consensus has been extensively utilized within industrial blockchains [38], [39] due to its higher performance [40]. Whereas the configuration of the blockchain network (i.e., block zero) can be found in the *Genesis.json* file with the source code. We also used *Node.js* and *Web3.js* libraries for the scripts detailing the interaction scenarios with the blockchain-based framework and SCs.

A. Performance Evaluation

The evaluation of our proposed scheme in terms of performance focuses on three main aspects: 1) the overhead introduced by the adoption of blockchain in terms of execution cost; 2) the scalability of the blockchain ledger with a growing number of devices; and 3) the runtime overhead associated to the additional utilized security mechanisms in RASSIFAB compared to a centralized scheme.

1) Execution and Communication Costs of the Initialization Phase: In order to assess the time required for setting up the framework (i.e., initialization and registration phase), we have tested the execution time (or cost) of various functions defined in the initialization SC developed with different workloads. We note that the execution time of a transaction triggering a function in the contract is defined as: $T_{\text{execution}} = T_{\text{confirmation}} - T_{\text{deployment}}$. Starting with adding the ANs to the system (i.e., TSO nodes), the obtained results in Table I indicate that it takes roughly 104.345 ms to add five ANs. Meanwhile, the initialization of the set of DSOs and OEMs takes 67.43 and 99.92 ms for adding three DSOs and five OEMs, respectively, to the blockchain network. Whereas the registration step of the nodes is conducted by adding ten DERANs and OEMNs for each of the added organizations (i.e., a total of 80 nodes), which requires 697.883 ms and 1.170 s, respectively.

Furthermore, the execution time of the registration phase of the devices (i.e., SIs) is also included in Table I. The procedure starts with updating the unique serial numbers of each SI part of the framework, which takes around 1.184 s for ten devices per manufacturer (i.e., a total of 50 devices). Then, adding the

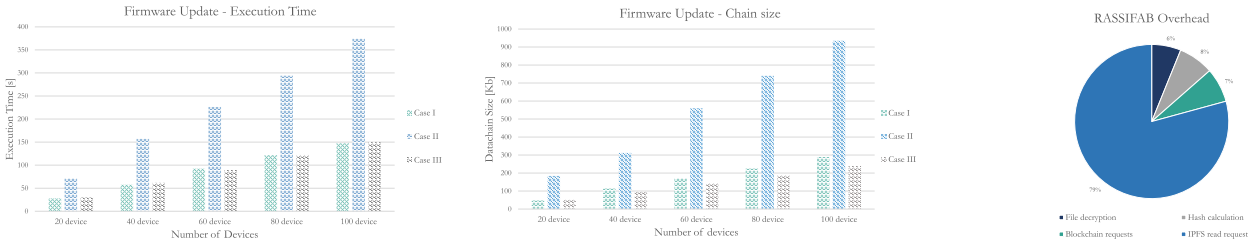

Fig. 7. Performance analysis of sending firmware updates using RASSIFAB. (a) Execution time. (b) Ledger scalability. (c) Runtime overhead.

TABLE I
TEST RESULTS OF RASSIFAB SCs DEPLOYMENT, INITIALIZATION, AND REGISTRATION PHASE

Functions	*Number*	*Exec. time [s]*	*Chaindata [kB]*
Registration of TSO nodes	5	0.104	6
Initialization of DSOs	3	0.067	3
Initialization of OEMs	5	0.099	6
Registration of DSOs nodes	10	0.697	27
Registration of OEMs nodes	10	1.170	41
Initialization of serial numbers	50	1.184	51
Registration of devices	50	1.220	52
Initialization of firmware metadata	50	1.246	71

Deployment time of *SIsInit.sol*: $23.957ms$
Deployment time of *SIsFirmware.sol*: $24.643ms$

blockchain addresses of the SIs and their respective associations to both the DSOs and OEMs requires 1.22 s. Finally, updating the mapping between the devices and their respective firmware metadata on-chain (i.e., the currently installed version on the SIs) is performed in 1.246 s.

Meanwhile, the communication cost of the scheme, which indicates the growth of the size of the chaindata as transactions are being added to the blockchain ledger was also evaluated. For instance, the registration of the DERANs and OEMNs takes 27 and 41 kB for each. Whereas the registration of the devices (i.e., addresses, serials, and their firmware metadata) takes a total average of 174 kB.

2) Execution Time of Sending Firmware Updates: The execution time of sending firmware updates to the devices within a single BZ, depicted in Fig. 7(a), was evaluated under three different scenarios. The first use case involves one single OEM part of the framework with four nodes (i.e., $|OEM| = 1$ and $|OEM_iN| = 4$). In the second case, we also have one OEM but with ten nodes (i.e., $|OEM| = 1$ and $|OEM_iN| = 10$). Whereas in the third case, the number of OEMs was set at five and to each four nodes were allocated (i.e., $|OEM| = 5$ and $|OEM_iN| = 4$; thus, $\sum_{i=1}^{5}\sum_{j=1}^{4} OEM_iN_j = 20$). It is important to note that we only present the evaluation of sending firmware updates here. Nonetheless, sending patches is roughly similar with the only exception regarding the requirement for the version number of the firmware within the metadata.

Starting with case I, the initial number of SIs was set at 20 and scaled to reach 100 devices that should be updated simultaneously. We should note that this number of devices represents the SIs within a single BZ, and as we mentioned previously, the whole LV network is divided into several zones, thus the total number of devices would be above 100. From Fig. 7(a), it can be observed that the time required for sending and validating the firmware update metadata on-chain for all cases is increasing with the number of devices which is quite expected. As more concurrent transactions are deployed to the blockchain, thus requiring more time to be confirmed, however, it is worth noting that the increase is linear and moderate. Meanwhile, the execution time of case II is noticeably higher compared to the other cases, which is due to the number of OEM nodes per manufacturer. The proposed scheme, as discussed previously, relies on a voting-like mechanism over the sent FOTA update metadata. Thus, in order to issue an event indicating that a new firmware is released, the metadata should be signed by at least six nodes in case II, compared to three nodes for the remaining cases (i.e., cases I and III). Nonetheless, the overhead introduced by blockchain in our proposed scheme for sending a firmware update request to a single device is around 1.093 s (in case I with at least three validating nodes), which is lower than the overhead in Hu et al. scheme [41] with an execution time of roughly 1.3 s using one single validating node.

3) Ledger Overhead With Sending Firmware Updates: The scalability of the chain size was also assessed as the number of SIs part of the system is increasing. From the obtained results depicted in Fig. 7(b), it can be observed that the ledger size increases as more transactions are added to the blockchain, yet the increase is following a linear trend also. For instance, sending a firmware update to 100 devices in case I adds roughly 289 kb to the chain, whereas in case II the ledger increases by 935 kb. It is worth stressing that the size of the ledger would grow in a moderate way, due to only saving the metadata of the firmware rather than the actual binary files, which are stored off-chain. In addition, the procedure of updating SIs is characterized by a low recurrence as typically a device can be expected to receive a firmware update twice a year on average. Thus, justifying the use of blockchain for keeping an immutable record of these security updates for future forensic analysis or security audits.

4) IPFS, Encryption, and Hash Runtime Overheads: The aim here is to evaluate the overhead introduced by all the

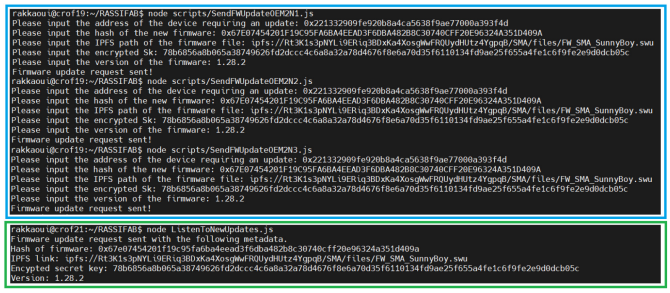

Fig. 8. Snapshot of sending a firmware update with no attacks.

mechanisms leveraged in our scheme compared to a centralized approach using the OEM's cloud servers with no requirements for files' encryption. If we consider a firmware binary image or file with a size equal to 16 MB, it would take approximately an extra 12 s to download the encrypted firmware from the IPFS, compared to using an HTTP server, following the benchmark provided in [42]. Then, 0.93 s to decrypt it using *AES-256* and 1.12 s to calculate the hash of the file using *SHA-256*, based on the benchmark conducted on a Raspberry Pi 3 model B, provided in [43]. Meanwhile, the time required to decrypt $S_{\text{key}}^{\text{Chain}}$ and compare the hashes are assumed to be negligible. Thus, the proposed scheme is expected to have an average runtime overhead of 15.143 s, which is within an acceptable range taking into account the new features introduced that harness the cybersecurity and resilience of the scheme. Besides, the procedure of firmware amendments is moderately sensitive to latency as it is vulnerable to *zero-day* attacks. In Fig. 7(c), we provide a pie-chart representing the average sum of RASSIFAB's runtime overhead that illustrates the various percentages of each additional technique utilized in our scheme.

B. Security Properties

In this section, we detail the various security features provided by RASSIFAB and discuss how they meet the goals defined in Section III-C with some test results.

1) Intrusion Prevention [$\mathcal{G}1$]: Fig. 8 represents the procedure of sending a firmware update (case III) under no attack to an SI that belongs to OEM_2. In order for the update request to be successful, the majority of the nodes (i.e., OEM_2N_1, OEM_2N_2, and OEM_2N_3 located in machine 3) need to send a transaction containing the metadata of the firmware. This procedure is depicted in the blue box of the figure, that represents the execution of three scripts to send a firmware update request from the OEM blockchain nodes. Only then would an event containing the new firmware metadata be automatically triggered at the SICN (located in machine 4) as illustrated in the green box of the figure. Our proposed scheme is capable of detecting various intrusion attacks that deviate from this detailed (normal) behavior, such as firmware downgrade attacks, unauthorized access, or malicious insiders in a fine-grained manner by triggering an exception sent to the blockchain nodes (e.g., ANs or OEMNs).

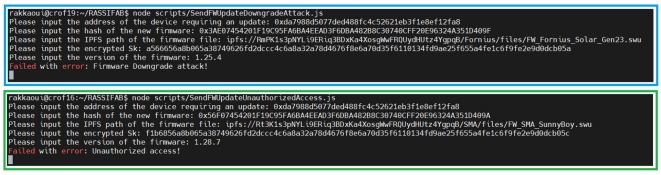

Fig. 9. Snapshots of two attack scenarios (i.e., downgrade attack and unauthorized access).

If we consider the attack scenario of a firmware downgrade attack, previously defined in Section III-B. For each request sent using the *sendFWupdate()* function, the version of the firmware is required as input. Then, the body of the function runs an autonomous check to verify that the provided version is higher than the previous one, which is recorded on-chain as part of the SIs' firmware metadata during the initialization phase. In the developed SC we used semantic versioning, a widely adopted version scheme based on three-part version number, where the first number indicates a major version change, the second for a minor and the third for patches. These numbers are assigned in increasing order and correspond to new developments in the firmware. An example of a firmware's version would be 1.7.23, which is converted into an array to be recorded on-chain as part of the firmware metadata. Suppose an attacker tries sending an old version of the firmware, the transaction will be reverted with an error message because it will not pass the predefined requirements of the function, as illustrated in Fig. 9 (i.e., blue box). The figure represents a snapshot of the execution of a script detailing an attack scenario in which an OEM node sends a request for an update with version 1.25.4, while the current firmware's version of the device is recorded as 1.27.3 on-chain. Nonetheless, it is possible that the attacker might input a higher version to bypass this check while the actual firmware is an old version. Still, the request will not be sent to the SICN as the metadata (i.e., firmware hash and IPFS path) need to be validated by the majority of the OEMNs, i.e., $SICN.Signers_{length} > |OEM_iN|/2 + 1$, which is impossible under the assumption of having the majority not malicious.

Meanwhile, if OEM_iN_j tries sending a firmware update to a device belonging to $OEM_{i'}$ with $i \neq i'$ (i.e., firmware mismatch attack), the transaction will be reverted with the error message: *Unauthorized access!*, as illustrated in Fig. 9 (i.e., the green box). The function requires the *msg.sender* to belong to the same organization as the device, this is achieved through the access control scheme we implemented in our proposed SCs. Another occurrence of this attack could be from the same OEM but for different models or types of devices, which could be due to an honest human error [37] or a malicious insider. In this case, as the new metadata need to be validated following a BFT protocol, the rest of the OEM nodes would notice this faulty update through logging and blockchain's events monitoring as illustrated in Fig. 10 (i.e., the gray box). Thus, the tampered metadata would never reach the SICN as it will not be validated by the rest of the legitimate OEM nodes, eventually protecting the device.

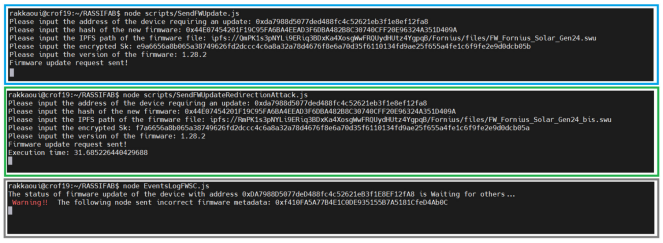

Fig. 10. Snapshot of an insider attack scenario being detected.

2) Firmware Integrity and Confidentiality [G2 and G3]: The integrity of the firmware means that it has not been tampered with during all steps of the FOTA procedure. In our scheme, we guarantee this feature by including the hashes of the binary files as well as the IPFS paths as part of the firmware metadata, which are recorded on the shared blockchain ledger among all entities part of the ecosystem. Due to the strong coalition resistance of hash functions, it would be impossible for an attacker to calculate the same hash for different files. Besides, the IPFS links are based on content identifiers (CID), where the link would have the following format: *ipfs*:// < *CID* > /*Path*/*file*. The CIDs are derived from the files' content hashes, thus any alteration of the content will produce a different CID, thus, a different path [15]. Suppose an adversary tries launching an MitM or redirection attack, the attacker would be unable to produce a different file with the same CID in the IPFS as illustrated in Fig. 10. This also applies to the case of a software supply chain attack where an adversary that was able to get access to the account of one of the OEMNs (e.g., through a leaked key or a malicious insider) is trying to send an update containing a malware similarly to SolarWinds attack. In this scenario, the attacker would be unable to generate a legitimate CID and hash of the corrupted firmware file. Besides, as the rest of the OEM nodes are: 1) assumed to be honest (i.e., at least the majority); 2) their accounts and blockchain keys were not leaked; and 3) the update requests in our scheme are required to be validated following a BFT manner, the attempted attack would be detected and the update request would fail.

Furthermore, as the firmware metadata are recorded in an immutable and distributed manner this enables the devices to check frequently whether their firmware has not been maliciously modified (e.g., in case of a physical attack). Meanwhile, as the proposed framework is based on a consortium blockchain and due to the transparency feature of the distributed system, the actual firmware files are encrypted before being pinned to the shared IPFS-based P2P network. Thus, guaranteeing the confidentiality of the firmware and protecting it from being leveraged for reverse engineering attacks.

3) Lightweight and Scalable FOTA [G4]: The proposed scheme alleviates the SICNs (considered as IoT devices with constrained capacities) from the burden of verifying each OEMN's signature while sending a firmware update. As this is encoded within the proposed *SIsFirmware.sol* SC and is executed automatically through the defined modifiers each time on-chain. Furthermore, the firmware binary images are encrypted using symmetric cryptography (i.e., AES-256), which has less computation overhead compared to asymmetric cryptography while still being fairly secure, fast and efficient.

Meanwhile, the scalability of the blockchain-based scheme is guaranteed by means of two approaches we adopted. The first is splitting the overall network of SIs into semiindependent zones, where each BZ would be responsible for processing a fraction of the transactional firmware updates/patches requests, thus increasing the overall throughput (through parallel processing) and minimizing the size of the light chain stored within the constrained SI devices for verification. Whereas the second approach achieves scalability by offloading the storage to the IPFS rather than keeping the actual firmware files within the blockchain network, it is also worth stressing that in our scheme we advise the use of an external shared P2P storage system for the firmware binary files. Nonetheless, RASSIFAB can still function using only the blockchain layer to keep an immutable record of the sent FOTA requests for updates.

VI. Related Work and Discussion

In this section, we discuss various research outputs in the literature that addressed the security concerns of FOTA amendments. By either relying on the traditional client-server model and enhancing it by leveraging strong cryptographic techniques as well as hardware-based mechanisms, or totally shifting to a distributed scheme by utilizing blockchain due to its immutability and inherent cyber-resilience, we also detail the limitations of our proposed blockchain-based framework. Furthermore, Table II represents a comparison between the discussed schemes and RASSIFAB in terms of various characteristics.

A. Centralized Hardware-Assisted Schemes

The procedure of remotely updating and/or patching firmware conventionally relies on a client-server architecture. Where the devices would either receive or retrieve the newly released firmware from the cloud server managed by the OEM of the IoT device. Several approaches in the literature have been proposed to secure this procedure by leveraging various cryptographic and/or hardware-assisted mechanisms. For instance, Kornaros et al. [44] introduced sCAN a secure scheme against cyberattacks targeting the CAN protocol utilized within vehicle networks, specifically against firmware reverse engineering attacks during FOTA, by means of additional signed metadata of the firmware and timestamps. The transfer of the firmware is based on a multiframe process, where the firmware file is divided into chunks that are verified by utilizing a cyclic redundancy check algorithm (i.e., checksum). Nonetheless, the proposed scheme still relies on the OEM's centralized cloud server to download the firmware files which might be the target of a malware injection attack. In addition, checksums do not necessarily guarantee the authenticity of the firmware as an insider attacker can change the data and recalculate the checksum to bypass the verification.

Meanwhile, ASSURED [45] is an architecture for firmware updates that guarantees end-to-end security between the OEMs

TABLE II
COMPARISON BETWEEN RASSIFAB AND OTHER FIRMWARE AMENDMENT SCHEMES

Ref.	Blockchain-based	Heterogeneous	BFT Verification	Distributed Repository	*Reputation*
ASSURED [45]	×	×	×	×	×
SecuCode [46]	×	×	×	×	×
Lee and Lee [49]	✓	×	–	✓	×
Hu *et al.* [41]	✓	×	×	✓	×
Yohan and Lo [48]	✓	✓	×	×	×
Choi and Lee [50]	✓	✓	×	✓	×
BoSMoS [51]	✓	×	×	×	×
RASSIFAB	✓	✓	✓	✓	✓

✓: *Satisfied* ×: *Unsatisfied* –: *Partially*

and the embedded IoT devices. The proposed framework introduces a controller (intermediary between the OEM and devices) responsible for approving the firmware metadata and transmitting the update envelope to the devices using an authenticated channel. However, the scheme relies on a static root-of-trust, i.e., the OEM and controller keys. Furthermore, Su et al. [46] presented SecuCode a secure wireless code dissemination scheme for computational radio frequency identification devices, which is built on the entanglement of a static random access memory PUF with a firmware update protocol. However, the scheme does not guarantee mutual authentication and is relying on a prover centralized database, initialized in a secure environment, which contains the IDs of the devices and their challenge–response pairs from the PUF.

Then, Rabbani et al. presented SHeFU [47], a secure firmware update for a homogenous network of IoT devices, that eliminates the requirement for remote attestation, where malicious devices that have been compromised by a faulty update are isolated from the rest of the network. Specifically, each device is responsible for calculating a firmware digest and upon the communication between the devices, each one would calculate a message authentication code for validation. However, the authors did not specify how the devices agree on which digest is legitimate. Moreover, the network owner in charge of monitoring the devices is assumed to be fully trusted and that it cannot be compromised, hence limiting the security of the scheme.

B. Distributed Blockchain-Enabled Schemes

In an attempt to address the flaws within the conventional client-server based schemes for FOTA, several researchers harnessed the potential of blockchain technology. For instance, in [48], Yohan and Lo proposed a blockchain-based scheme to guarantee the authenticity and integrity of FOTA updates within a heterogeneous IoT devices network. The procedure follows a push approach and can be performed either directly or indirectly. In the direct FOTA mechanism, vendors create and deploy the corresponding SCs to the blockchain network for each firmware update. Whereas the indirect scheme introduces a firmware broker that serves as an intermediary between the vendor and the targeted IoT devices. However, the scheme still relies on the centralized repositories of either the vendor or the broker. In addition, after downloading the firmware, there is no mechanism implemented to verify that the file or image was not compromised during the transfer. Furthermore, the scheme requires that each firmware update should be associated with a newly deployed SC to the blockchain, which is not sustainable taking into account the ledger scalability and overhead introduced over the long run.

Meanwhile, in the blockchain-based scheme proposed by Lee and Lee [49], embedded devices request their firmware updates to the P2P network, to get a response checking whether the firmware is up-to-date. The framework is based on three types of nodes, i.e., normal nodes, verification nodes and a vendor node (which is outside the blockchain system but used by the verification nodes in order to provide the file updates). Normal nodes can be either request or response nodes. The procedure starts by having a normal node sending a version-check request, broadcasted to the whole network, and depending on the node receiving the request the scheme follows different verification steps. However, the procedure is tailored for homogenous IoT-based networks belonging to the same vendor, as each device would compare its current version with the other peers, which does not describe accurately the reality of IoT-based ecosystems supporting different devices from various OEMs. Furthermore, because of the nature of the broadcasted request the scheme is susceptible to create repetitive operations. For instance, the sent request can be verified by a verification node, but a response node might also conduct a confirmation using proof-of-work which is regarded as an extensive consensus mechanism, thus consuming more unnecessary network traffic and computational resources. In addition, when the embedded device is required to download a newly released firmware, it requests a list of peers from where it can fetch the firmware file. However, as the operation is performed only by the verification node without consensus from the rest of the peers, it is plausible that the device requiring an update falls to an eclipse attack where a malicious verification node could try to send a compromised peers' list.

Furthermore, Hu et al. [41] proposed an autonomous scheme for FOTA updates with malware check. The procedure is based on an SC that records the firmware metadata on-chain and leverages several off-chain programs to validate the correctness of the sent data by the manufacturer node. However, no implementation details were provided regarding the off-chain verification procedure and how the blockchain nodes come to consensus before indicating that the request to record the metadata of a new firmware is reliable. In addition, the malware scan ratio part of the firmware metadata is only uploaded by a single manufacturer node without any verification by other peers, thus, if one of these nodes is hacked an attacker could send a corrupted file that was intelligently reverse engineered to bypass the external virus check. Meanwhile, the on-chain verification is only based on checking the signature of the manufacturer node as well as the provided score from the external malware check tool, which does not necessarily guarantee the

reliability of the firmware in case of leaked keys or insider attacks.

Whereas in [50], Choi and Lee proposed a distributed FOTA architecture design for IoT devices based on blockchain. The scheme leverages manifests defined in SUIT [28], that are recorded on the immutable ledger by a vendor (or author as referred to in this article). The architecture offers a solution to the author-disappearance concern, where the server of an OEM is targeted by an attack hindering the devices from downloading the latest firmware update or patch. Nonetheless, the scheme introduces a retrieval node (a trusted gateway between the IoT devices and the blockchain network) which could represent a security threat. Moreover, the procedure of checking the firmware and its metadata is solely based on the provided data by the author node and its digital signature. Thus, if the node's key is leaked an attacker could send a malicious firmware with a modified manifest to bypass the checksum verification. Furthermore, the IoT devices are required to send a transaction periodically to query if there is a new firmware, which might add up to the ledger's size if the blockchain system encompasses thousands of devices. Furthermore, the scheme lacks a performance evaluation to assess its feasibility and efficacy as it has not been implemented.

Besides, in BoSMoS [51], He et al. focused on the validity of IoT devices' software rather than the procedure of issuing updates or patches. They designed a blockchain-enabled software status monitoring framework in order to detect any malicious attempts to compromise the devices and respond to the intrusion. The system is based on taking snapshots of the software, using a trusted security monitoring module, which are then stored on the immutable blockchain ledger for periodic verification. Nonetheless, the scheme also relies on the trusted software snapshots that are updated by a single developer and signed using its public key to then be registered on the blockchain ledger. In case the private key of the developer is leaked or compromised, this could jeopardize the security of the monitoring system and integrity of the recorded software metadata on-chain. Meanwhile, the IoT devices within the scheme are not assigned with keys to interact with the blockchain, but they rely solely on a gateway that aggregates all blockchain requests from these devices and signs them with its key. Thus, the scheme fails to provide a certain degree of auditability (i.e., in terms of incoming requests from the IoT devices, for instance to determine the root of a denial of service attack on the blockchain system). In addition, the design of the framework is more fit for homogenous IoT devices and the authors did not address the case where the system encompasses various OEMs with different access rules.

C. Advantages and Limitations of RASSIFAB

On the one hand, an FOTA scheme can rely on the strong cryptography protocols provided by blockchain (i.e., ECDSA, SHA, etc.) to guarantee the immutability and nonrepudiation of the firmware's files and their metadata. However, if we fail to verify the recorded data on-chain by relying on a single-root of trust (e.g., OEM author), several things could go wrong. In fact, if we start from the presumption that all manufacturer nodes are trusted in the blockchain-based ecosystem and that the rest of the validators need only to verify the transactions' signatures of these nodes, we might fail at achieving our required security goals (i.e., the integrity of the firmware amendment requests and their metadata). Thus, our scheme incorporates a voting-based verification mechanism for the uploaded firmware's metadata, rather than the actual binary files for efficiency and scalability, yet without compromising on the security aspect of the scheme. Furthermore, RASSIFAB also incorporates an immutable and auditable reputation scheme used to mandate the access control of the vendors' node parts of the FOTA framework. Where reputation scores are updated in an automated manner using SCs and based on the continuous behavior of these nodes within the blockchain network. Nonetheless, it is worth acknowledging that classifying misbehavior to assign new reputation scores as well as ensuring the reliability of these scores across zones is a challenging task that was the subject of some research outputs [52], [53] and could be the focus of future work. Besides, for the storage of the firmware files we proposed to use the IPFS to form a consortium system among all OEMs where the binary files would be encrypted before being pinned to the P2P systems, hence guaranteeing the confidentiality of the manufacturers' proprietary code. Nonetheless, the protocol is still in its infancy and distributed storage, in general, is still an ongoing line of research with several challenges that remain to be addressed (e.g., free riders, incentives, etc.) [15]. Moreover, the framework was implemented on only four machines, which might not reflect a real case scenario. Nonetheless, RASSIFAB is based on the Ethereum platform where its public implementation supports up to ten thousand physical nodes scattered around several countries. Also, our scheme leverages *Clique*, which has a higher throughput and minimized latency with a growing network size (i.e., number of nodes) compared to other consensus mechanisms [40].

On the other hand, RASSIFAB only focuses on the security of software supply chain rather than hardware supply chain. For the latter, several blockchain-based schemes exist in the literature. For instance, Xu et al. [54] proposed the design of a blockchain-based framework used to manage critical information about chips and to mitigate the risk of components recycling, cloning, overproduction, etc. Similarly, Cui et al. [55] also addressed the hardware supply chain provenance concern by proposing a Hyperledger-based framework used to ensure the traceability of electronic components as they are circulating between manufacturers, distributors and end users. Guin et al. [56] also used blockchain to ensure an authentic tracking of the origin of edge devices by leveraging the PUF for authentication. In a nutshell, hardware supply chain schemes are used at the early beginning of a device's lifecycle to ensure the authenticity and provenance of all its components. This aspect is extremely important in the case of SIs as those are also composed of several chips and their origins as well as integrity during the manufacturing or assembling procedures are critical. We acknowledge that eluding the security of hardware supply chain might limit the capabilities of our proposed framework, but we also would like to note that this matter falls beyond the scope of this article and

could be investigated in the future. For instance, the hardware provenance of the SIs would be coupled with the software maintenance provided in our framework, and the PUF would offer a strong authentication means for the SIs. Last, we should note that ensuring the integrity of SIs' firmware amendments would guarantee a partial security of the whole ecosystem, as attackers could still launch a top-down attack from the aggregators or DSOs' DERMSs controlling these inverters to compromise their primary settings. In this case, our proposed blockchain-based framework can be extended in the sense of also incorporating the control aspect (e.g., voltage or frequency control) of these DERs equipped with their IoT-enabled SIs.

VII. Conclusion and Future Directions

In this article, we presented the design of RASSIFAB, which is an FOTA scheme based on blockchain technology to guarantee a secure, resilient, reliable, and auditable procedure of sending updates and/or patches to SIs connected to the LV networks of the SG within residential areas. The framework enables the deployment of large-scale DERs with heterogeneous SIs, by leveraging network segmentation and blockchain sharding. Thus, increasing the overall throughput of the blockchain system in terms of processed transactions and minimizing attack entry points from a cybersecurity perspective. The procedure of verifying the sent requests for updates by the OEM nodes is performed following a BFT manner rather than starting from the assumption of having trusted OEM nodes, which we viewed as unrealistic due to the threat of insider attacks or simply human error. Last but not least, the proposed framework was intended to be used for SIs; however, we should note that it was designed in a generic and modular way that would enable it to manage also the procedure of FOTA updates for other IoT-enabled devices within the SG or other industrial applications.

For future work, we intend to study the collaborative aspect between the various blockchain zones in terms of intrusion attacks. For instance, if one OEM node is blacklisted within a given zone, how can we ensure the correctness and fairness of denying his access to the others. In addition, our scheme was only tested on several machines that were partitioned to emulate the resources of the devices running as light nodes. Nonetheless, the possibility of configuring and running a blockchain client on a Raspberry Pi, that would then be interfaced with a real SI to perform the verification check over the firmware and its metadata is also worth investigating. Last, integrating RASSIFAB with other blockchain-based hardware supply chain solutions to guarantee the authenticity of the SIs using their PUF and DERs control schemes could also be the focus of future research.

References

[1] (Int. Energy Agency, Paris, France). *Renewables 2020 Report*. (Nov. 2020). Accessed: Oct. 2, 2023. [Online]. Available: https://www.iea.org/reports/renewables-2020

[2] "*IEEE Standard For Interconnection and Interoperability of Distributed Energy Resources With Associated Electric Power Systems Interfaces*," IEEE Standard 1547-2018, Apr. 2018. Accessed: Oct. 2, 2023. [Online]. Available: https://standards.ieee.org/ieee/1547/5915/

[3] "Power generating plants in the low voltage network," Berlin, Germany, VDE FNN, document VDE-AR-N 4105, Apr. 2019. Accessed: Oct. 2, 2023. [Online]. Available: https://www.vde.com/en/fnn/topics/technical-connection-rules/power-generating-plants

[4] M. Antonakakis et al., "Understanding the mirai botnet," in *Proc. 26th USENIX Security Symp.*, Aug. 2017, pp. 1093–1110.

[5] C. Xenofontos, I. Zografopoulos, C. Konstantinou, A. Jolfaei, M. K. Khan, and K. R. Choo, "Consumer, commercial, and Industrial IoT (In)Security: Attack taxonomy and case studies," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 199–221, Jan. 2022, doi: 10.1109/JIOT.2021.3079916.

[6] S. Soltan, P. Mittal, and H. V. Poor, "BlackIoT: IoT botnet of high wattage devices can disrupt the power grid," in *Proc. 27th USENIX Security Symp.*, Aug. 2018, pp. 15–32.

[7] J. Ye et al., "A review of cyber–physical security for photovoltaic systems," *IEEE J. Emerg. Sel. Topics Power Electron.*, vol. 10, no. 4, pp. 4879–4901, Aug. 2022, doi: 10.1109/JESTPE.2021.3111728.

[8] J. Johnson, J. Quiroz, R. Concepcion, F. Wilches-Bernal, and M. J. Reno, "Power system effects and mitigation recommendations for DER cyber-attacks," *IET Cyber-Phys. Syst. Theory Appl.*, vol. 4, no. 3, pp. 240–249, Sep. 2019, [Online]. Available: https://doi.org/10.1049/iet-cps.2018.5014

[9] G. Tertytchny et al., "Demonstration of man in the middle attack on a commercial photovoltaic inverter providing ancillary services," in *Proc. IEEE CyberPELS*, Miami, FL, USA, Jan. 2021, pp. 1–7, doi: 10.1109/CyberPELS49534.2020.9311531.

[10] D. E. Whitehead, K. Owens, D. Gammel, and J. Smith, "Ukraine cyber-induced power outage: Analysis and practical mitigation strategies," in *Proc. 70th Annu. CPRE*, College Station, TX, USA, Nov. 2017, pp. 1–8, doi: 10.1109/CPRE.2017.8090056.

[11] C. Konstantinou and M. Maniatakos, "Impact of firmware modification attacks on power systems field devices," in *Proc. IEEE Int. Conf. SmartGridComm*, Miami, FL, USA, Mar. 2016, pp. 283–288, doi: 10.1109/SmartGridComm.2015.7436314.

[12] (MITRE, McLean, VA, USA). *CVE-2017-9860*. Accessed: Oct. 2, 2023. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-9860

[13] "Thales–2022 thales cloud security study: The challenges of data protection in a multicloud world." 2022. Accessed: Oct. 2, 2023. [Online]. Available: https://cpl.thalesgroup.com/cloud-security-research#download-popup

[14] Md N. Islam and S. Kundu, "Enabling IC traceability via blockchain pegged to embedded PUF," *ACM Trans. Design Autom. Electron. Syst.*, vol. 24, no. 36, pp. 1–23, Apr. 2019. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1145/3315669

[15] E. Daniel and F. Tschorsch, "IPFS and friends: A qualitative comparison of next generation peer-to-peer data networks," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 1, pp. 31–52, 1st Quart., 2022, doi: 10.1109/COMST.2022.3143147.

[16] R. Dhobi, S. Gajjar, D. Parmar, and T. Vaghela, "Secure firmware update over the air using trustzone," in *Proc. IEEE i-PACT*, Vellore, India, Jan. 2020, pp. 1–4, doi: 10.1109/i-PACT44901.2019.8959992.

[17] K. Suzaki, A. Tsukamoto, A. Green, and M. Mannan, "Reboot-oriented IoT: Life cycle management in trusted execution environment for disposable IoT devices," in *Proc. ACSAC*, New York, NY, USA, Dec. 2020, pp. 428–441. [Online]. Available: https://doi.org/10.1145/3427228.3427293

[18] M. A. Prada-Delgado, A. Vázquez-Reyes, and I. Baturone, "Trustworthy firmware update for Internet-of-Thing devices using physical unclonable functions," in *Proc. IEEE GIoTS*, Geneva, Switzerland, Aug. 2017, pp. 1–5, doi: 10.1109/GIOTS.2017.8016282.

[19] D. Mbakoyiannis, O. Tomoutzoglou, and G. Kornaros, "Secure over-the-air firmware updating for automotive electronic control units," in *Proc. 34th ACM/SIGAPP Symp. Appl. Comput.*, Limassol, Cyprus, Apr. 2019, pp. 174–181. [Online]. Available: https://doi.org/10.1145/3297280.3297299

[20] B.-C. Choi, S.-H. Lee, J.-C. Na, and J.-H. Lee, "Secure firmware validation and update for consumer devices in home networking," *IEEE Trans. Consum. Electron.*, vol. 62, no. 1, pp. 39–44, Feb. 2016, doi: 10.1109/TCE.2016.7448561.

[21] D. Cooper et al., "Security considerations for code signing," Gaithersburg, MD, USA, NIST, NIST Cybersecurity White Paper, Jan. 2018. Accessed: Oct. 2, 2023. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.01262018.pdf

[22] W. Viriyasitavat, L. D. Xu, Z. Bi, and D. Hoonsopon, "Blockchain technology for applications in Internet of Things—Mapping from system design perspective," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8155–8168, Oct. 2019, doi: 10.1109/JIOT.2019.2925825.

[23] M. A. Ferrag and L. Shu, "The performance evaluation of blockchain-based security and privacy systems for the Internet of Things: A tutorial," *IEEE Internet Things J.*, vol. 8, no. 24, pp. 17236–17260, Dec. 2021, doi: 10.1109/JIOT.2021.3078072.

[24] A. Pillai, M. Sindhu, and K. V. Lakshmy, "Securing firmware in Internet of Things using blockchain," in *Proc. 5th ICACCS*, Coimbatore, India, Jun. 2019, pp. 1–6, doi: 10.1109/ICACCS.2019.8728389.

[25] R. Akkaoui, "Blockchain for the management of Internet of Things devices in the medical industry," *IEEE Trans. Eng. Manag.*, vol. 70, no. 8, pp. 2707–2718, Aug. 2023, doi: 10.1109/TEM.2021.3097117.

[26] G. Bere, B. Ahn, J. J. Ochoa, T. Kim, A. A. Hadi, and J. Choi, "Blockchain-based firmware security check and recovery for smart inverters," in *Proc. IEEE APECE*, Phoenix, AZ, USA, Jul. 2021, pp. 1–5, doi: 10.1109/APEC42165.2021.9487468.

[27] H. Tschofenig and S. Farrell, "Report from the Internet of Things software update (IoTSU) workshop 2016," IETF, Fremont, CA, USA, Rep. RFC 8240, Sep. 2017. Accessed: Oct. 2, 2023. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8240.html

[28] "Software updates for Internet of Things (SUIT) working group." 2017. Accessed: Oct. 2, 2023. [Online]. Available: https://datatracker.ietf.org/wg/suit/about/

[29] B. Moran, H. Tschofenig, D. Brown, and M. Meriac, "A firmware update architecture for Internet of Things," IETF, Fremont, CA, USA, Rep. RFC 9019, Apr. 2021. Accessed: Oct. 2, 2023. [Online]. Available: https://datatracker.ietf.org/doc/rfc9019/

[30] R. Housley, 'Using cryptographic message syntax (CMS) to protect firmware packages," IETF, Fremont, CA, USA, Rep. RFC 4108, Jan. 2020. Accessed: Oct. 2, 2023. [Online]. Available: https://datatracker.ietf.org/doc/rfc4108/

[31] "Security for industrial automation and control systems–Part 2-3: Patch management in the IACS environment." IEC, Geneva, Switzerland, IEC Rep. TR 62443-2-3, Jun. 2015. Accessed: Oct. 2, 2023. [Online]. Available: https://webstore.iec.ch/publication/22811

[32] *SolarWinds Cyberattack Demands Significant Federal and Private-Sector Response*, U.S. Govt. Accountabil. Office, Washington, DC, USA, Apr. 2021. Accessed: Oct. 2, 2023. [Online]. Available: https://www.gao.gov/blog/solarwinds-cyberattack-demands-significant-federal-and-private-sector-response-infographic

[33] *Draft Guide for Cybersecurity of Distributed Energy Resources Interconnected With Electric Power Systems*, IEEE Standard P1547.3, 2023. Accessed: Oct. 2, 2023. [Online]. Available: https://standards.ieee.org/ieee/1547.3/10173/

[34] *Inverters, Converters, Controllers and Interconnection System Equipment for Use With Distributed Energy Resources*, UL Standard 1741, Sep. 2021. Accessed: Oct. 2, 2023. [Online]. Available: https://standardscatalog.ul.com/ProductDetail.aspx?productId¯UL1741#

[35] S. Brewster. "UL and NREL announce cybersecurity testing recommendations for distributed energy resources and inverter based resources." Mar. 2022. Accessed: Oct. 2, 2023. [Online]. Available: https://www.ul.com/news/ul-and-nrel-announce-cybersecurity-testing-recommendations-distributed-energy-resources-and

[36] A. Shostack, *Threat Modeling: Designing for Security, Chapter 3: STRIDE*. New York, NY, USA: Wiley, Feb. 2014.

[37] S. Kok. "Faulty update brainwashes AEG microwaves into thinking they're steam ovens." Mar. 2022. Accessed: Oct. 2, 2023. [Online]. Available: https://thenextweb.com/news/update-brainwashes-microwaves-thinking-theyre-steam-ovens

[38] J. Yang, J. Dai, H. B. Gooi, H. D. Nguyen, and A. Paudel, "A proof-of-authority blockchain-based distributed control system for islanded microgrids," *IEEE Trans. Ind. Informat.*, vol. 8, no. 11, pp. 8287–8297, Nov. 2022, doi: 10.1109/TII.2022.3142755.

[39] Y. Pang, D. Wang, X. Wang, J. Li, and M. Zhang, "Blockchain-based reliable traceability system for telecom big data transactions," *IEEE Internet Things J.*, vol. 9, no. 14, pp. 12799–12812, Jul. 2022, doi: 10.1109/JIOT.2021.3138462.

[40] A. Ahmad, A. Alabduljabbar, M. Saad, D. Nyang, J. Kim, and D. Mohaisen "Empirically comparing the performance of blockchain's consensus algorithms," *IET Blockchain*, vol. 1, no. 1, pp. 56–64, May 2021. [Online]. Available: https://doi.org/10.1049/blc2.12007

[41] J.-W. Hu, L.-Y. Yeh, S.-W. Liao, and C.-S. Yang, "Autonomous and malware-proof blockchain-based firmware update platform with efficient batch verification for Internet of Things devices," *Comput. Security*, vol. 86, pp. 238–252, Sep. 2019. [Online]. Available: https://doi.org/10.1016/j.cose.2019.06.008

[42] J. Shen, Y. Li, Y. Zhou, and X. Wang, "Understanding I/O performance of IPFS storage: A client's perspective," in *Proc. IEEE/ACM 27th IWQoS)*, Apr. 2020, pp. 1–10, doi: 10.1145/3326285.3329052.

[43] M. El-Hajj, M. Chamoun, A. Fadlallah, and A. Serhrouchni, "Analysis of cryptographic algorithms on IoT hardware platforms," in *Proc. 2nd CSNet*, Paris, France, 2019, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/8602942

[44] G. Kornaros et al., "Towards holistic secure networking in connected vehicles through securing CAN-bus communication and firmware-over-the-air updating," *J. Syst. Archit.*, vol. 109, Oct. 2020, Art. no. 101761. [Online]. Available: https://doi.org/10.1016/j.sysarc.2020.101761.

[45] N. Asokan, T. Nyman, N. Rattanavipanon, A. Sadeghi, and G. Tsudik, "ASSURED: Architecture for secure software update of realistic embedded devices," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2290–2300, Nov. 2018, doi: 10.1109/TCAD.2018.2858422.

[46] Y. Su, Y. Gao, M. Chesser, O. Kavehei, A. Sample, and D. C. Ranasinghe, "SecuCode: Intrinsic PUF entangled secure wireless code dissemination for computational RFID devices," *IEEE Trans. Depend. Secure Comput.*, vol. 8, no. 4, pp. 1699–1717, Jul./Aug. 2021, doi: 10.1109/TDSC.2019.2934438.

[47] M. M. Rabbani, J. Vligen, M. Conti, and N. Mentens, "SHeFU: Secure hardware-enabled protocol for firmware updates," in *Proc. IEEE ISCAS*, Seville, Spain, 2020, pp. 1–5, doi: 10.1109/ISCAS45731.2020.9180850.

[48] A. Yohan and N. W. Lo, "An over-the-blockchain firmware update framework for IoT devices," in *Proc. IEEE Conf. DSC*, Kaohsiung, Taiwan, Jan. 2019, pp. 1–8, doi: 10.1109/DESEC.2018.8625164.

[49] B. Lee and J. H. Lee, "Blockchain-based secure firmware update for embedded devices in an Internet of Things environment," *J. Supercomput.*, vol. 73, pp. 1152–1167, Sep. 2016. [Online]. Available: https://doi.org/10.1007/s11227-016-1870-0

[50] S. Choi and J. H. Lee, "Blockchain-based distributed firmware update architecture for IoT devices," *IEEE Access*, vol. 8, pp. 37518–37525, 2020, doi: 10.1109/ACCESS.2020.2975920.

[51] S. He, W. Ren, T. Zhu, and K. R. Choo, "BoSMoS: A blockchain-based status monitoring system for defending against unauthorized software updating in industrial Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 948–959, Feb. 2020, doi: 10.1109/JIOT.2019.2947339.

[52] X. Xu, J. Gu, H. Yan, W. Liu, L. Qi, and X. Zhou, "Reputation-aware supplier assessment for blockchain-enabled supply chain in industry 4.0," *IEEE Trans. Ind. Informat.*, vol. 19, no. 4, pp. 5485–5494, Apr. 2023, doi: 10.1109/TII.2022.3190380.

[53] Y. He, C. Zhang, B. Wu, Y. Yang, K. Xiao, and H. Li, "A cross-chain trusted reputation scheme for a shared charging platform based on blockchain," *IEEE Internet Things J.*, vol. 9, pp. 7989–8000, Jun. 2022, doi: 10.1109/JIOT.2021.3099898.

[54] X. Xu, F. Rahman, B. Shakya, A. Vassilev, D. Forte, and M. Tehranipoo4, " Electronics supply chain integrity enabled by blockchain," *ACM Trans. Design Autom. Electron. Syst.*, vol. 24, no. 3, pp. 1–25, May 2019. [Online]. Available: https://doi.org/10.1145/3315571.

[55] P. Cui, J. Dixon, U. Guin, and D. Dimase, "A blockchain-based framework for supply chain provenance," *IEEE Access*, vol. 7, pp. 157113–157125, 2019, doi: 10.1109/ACCESS.2019.2949951.

[56] U. Guin, P. Cui, and A. Skjellum, "Ensuring proof-of-authenticity of IoT edge devices using blockchain technology," in *Proc. IEEE Int. Conf. Blockchain*, Halifax, NS, Canada, 2018, pp. 1042–1049, doi: 10.1109/Cybermatics_2018.2018.00193.

Raifa Akkaoui received the M.Eng. degree from the National Institute of Posts and Telecommunications, Rabat, Morocco, in 2016, and the Ph.D. degree from Huazhong University of Science and Technology, Wuhan, China, in 2020.

In 2021, she joined Delft University of Technology, Delft, The Netherlands, as a Postdoctoral Researcher. Her research interests include blockchain, edge computing, Internet of Things, cybersecurity, smart grids, and game theory.

Alexandru Stefanov (Member, IEEE) received the M.Sc. degree from the Politehnica University of Bucharest, Bucureşti, Romania, in 2011, and the Ph.D. degree from University College Dublin, Dublin, Ireland, in 2015.

He is an Assistant Professor of Intelligent Electrical Power Grids with Delft University of Technology, Delft, The Netherlands, and the Scientific Director of the Control Room of the Future Technology Centre, Department of Electrical Sustainable Energy. He holds the professional title of a Chartered Engineer from Engineers Ireland. His research interests are cybersecurity for power grids, resilience of cyber–physical systems, and next-generation grid operation.

Peter Palensky (Senior Member, IEEE) received the M.Sc. degree in electrical engineering and the Ph.D. and Habilitation degrees from Vienna University of Technology, Vienna, Austria, in 1997, 2001, and 2015, respectively.

He cofounded Envidatec, Hamburg, Germany, a German startup on energy management and analytics. In 2008, he joined the Lawrence Berkeley National Laboratory, Berkeley, CA, USA, as a Researcher, and the University of Pretoria, Pretoria, South Africa. In 2009, he became appointed as the Head of the Business Unit on Sustainable Building Technologies, Austrian Institute of Technology (AIT), Vienna, and later, the first Principal Scientist of Complex Energy Systems with AIT. In 2014, he was appointed as a Full Professor of Intelligent Electric Power Grids with Delft University of Technology, Delft, The Netherlands. His research interests include energy automation networks, smart grids, and modeling intelligent energy systems.

Prof. Palensky is active in international committees, such as ISO or CEN. He also serves as an IEEE IES AdCom Member-at-Large for various functions for IEEE. He is the past Editor-in-Chief of the *IEEE Industrial Electronics Magazine*, and an Associate Editor of several other IEEE publications, and regularly organizes IEEE conferences.

Dick H. J. Epema received the M.Sc. degree in mathematics with a minor in Spanish and the Ph.D. degree in mathematics (algebraic geometry) from Leiden University, Leiden, The Netherlands, in 1979 and 1983, respectively, and the M.Sc. degree in computer science from Delft University of Technology, Delft, The Netherlands, in 1988.

He is an Emeritus Professor of Distributed Systems with Delft University of Technology, Delft, The Netherlands. He has authored more than 140 scientific papers and has been on numerous program committees in grids, clouds, and P2P computing. His research interests are in the areas of resource management in distributed systems and in blockchain technology.

Prof. Epema was an Associate Editor of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS and IEEE TRANSACTIONS ON CLOUD COMPUTING. He was the General Co-Chair of EuroPar 2009 and IEEE P2P 2010, and the General Chair of HPDC 2012 and CCGrid 2013. He was the Program Committee Co-Chair of HPDC 2013.