An Investigation into Secure, Remote, Firmware Updating Mechanisms for Peer-to-Peer Transactive Microgrids

Andrew C. Smith
Next Generation Enterprises and Institutions
Council for Scientific and Industrial Research
Pretoria, South Africa
acsmith@csir.co.za

Lehlogonolo P.I. Ledwaba
Next Generation Enterprises and Institutions
Council for Scientific and Industrial Research
Pretoria, South Africa
lledwaba4@csir.co.za
0000-0002-7292-2835

***Abstract*—Renewable energy-based microgrid deployments are being identified as potential solutions for faster electrification in developing countries. IoT-enabled microgrids solve the physical infrastructure limitations of connecting communities that are geographically distant from main grid energy supply networks and reduce the added demand placed on the already grid. However, with the deployment of long-term, long-lived IoT technologies, a need for appropriate maintenance and updating strategies is introduced to ensure that the network's security, integrity, and availability is maintained. Hardware would need to access up-to-date features and patches deployed within newer firmware iterations without significant interaction and effort from the end user.**

This work aims to identify, evaluate, and recommend appropriate strategies and solutions for remote IoT firmware updating to be used within transactive microgrid deployments. The solutions considered should be able to maintain the security and integrity of the firmware file during distribution and be able to tolerate the unpredictability of transmission utilizing various communications networks and differing levels of network coverage. The investigation compares and analyses various firmware updating methodologies for lightweight operation, capability of minimising the monetary cost of firmware updating to the end user, and coverage of firmware updating attack vectors

As part of future work, the identified firmware mechanisms shall be implemented within a demonstrable microgrid network simulation to assess the performance and latency impacts introduced on microgrid transactions and IoT network processes.

***Index Terms*—Firmware updates, Transactive Microgrids, Security**

I. Introduction

The move towards distributed energy generation as a solution to increasing energy demands has seen a rise in transactive microgrid solutions. The microgrid is highly complex and is comprised of a multitude of systems from distributed energy resources (DERS), energy management systems (EMS) and transactive energy markets (TEM). One of the main enabling technologies within the microgrid is the Internet of Things which covers the breadth of smart devices used in the design and realisation of the microgrid network. The devices can range from highly low-power, resource constrained edge sensor and actuator devices for the monitoring of energy generation infrastructure to more high powered, fog layer gateway devices handling community level energy trading and load balancing. With the highly interconnected nature of the data sharing and on-going communications requires within the transactive microgrid, ensuring the devices are protected against vulnerabilities and security attack is imperative as the distributed nature of the network makes traditional cybersecurity and monitoring difficult to implement; especially with large scale deployments, One of the main ways in which IoT devices can remain protected against vulnerability is through consistent firmware updating and patching.

Remote firmware updates for IoT devices is a broad topic, potentially covering issues such as bandwidth, electrical power, processing power, timing, security (confidentiality, integrity, and availability), and many other dimensions. Updating of IoT devices is tricky owing to the variety of devices that can be found within a network deployment, the differences in capability that could impact the devices ability to implement the update, the availability requirements of the device within the network as well as the ICT and communications infrastructure that may be avaiable for an update to be pushed to a device. To illustrate the difference in capabilities that a firmware update would need to consider, Figure 1 illustrates some of the most common devices found within the transactive microgrid classified according to the device class categories developed by El Jaouhari and Bouvet [1]. Class 1-2 devices would not be able to handle firmware updates in the same manner as Class 5 devices therefore an effective strategy to manage updates throughout the microgrid would need to be established.

Within the scope of this literature review, certain assumptions are made and described in the following scenario; focusing on remote update mechanisms and explicitly excluding mechanisms such as UART, USB, microSD, and Flash. This

This work was supported in part by the Council of Scientific and Industrial Research, South Africa and the Department of Science and Innovation, South Africa within the ambit of the Foundational Digital Capabilities Research (FDCR) programme [Project KR5ETRT]

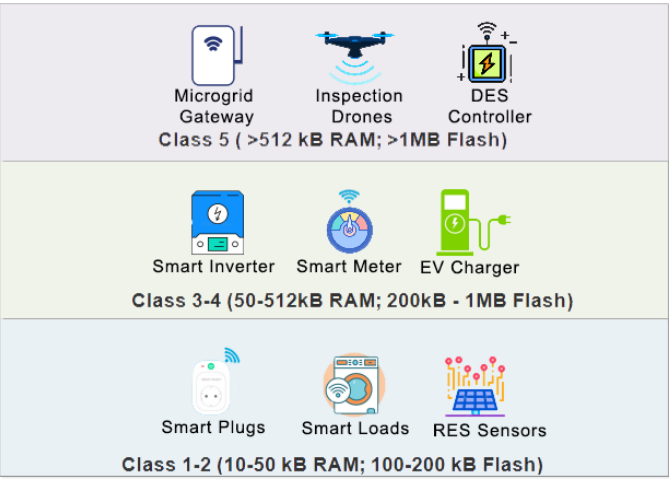

Fig. 1. IoT Devices with the Transactive Microgrid

is keeping with previous work in which existing choices were made towards the design and development of a transactive, peer-to-peer microgrid for the South African context (both rural and urban deployments) [2].

This study analyses remote firmware updating mechanisms for Class 3-5 edge node devices with processing power and RAM at a minimum of a Raspberry Pi 3B, a higher-end device compared to a resource constrained IoT device [3]. Firmware updating techniques for low-power, resource constrained Class 1 devices shall be considered as part of future work.

It is also assumed that the communication bandwidth is low, like LPWAN with 2G/3G at a maximum. The edge node is therefore considered to be a device operating in a bandwidth-constrained environment, not because of its own limitations, but because of the network to which it is connected. Thus, the HTTP protocol is not suitable [4] for the use case considered here. Subsequently, RFC9019, which is based on HTTP, is also not applicable in this case.

Also, assume that the number of geographically separated edge nodes are in the order of millions. This has an impact on the firmware image distribution mechanism since a single centralized point of distribution is not efficient in this case [5]. Furthermore, based on Kohout et al.'s [6] assertions, assume the design must cater for an in-use life span more than 15 years.

Finally, assume that no user interface exists, that no human is involved in the update process, and that no user consent is required to update the firmware. Therefore, a mechanism supporting unattended updates is required.

The rest of this work is organised as follows: Section II gives a more detailed background on the need for remote firmware updating for the transactive microgrid as well as the associated security requirements, Section III looks at some of the updating models in which devices are able to receive firmware update images while Section IV looks into some of the mechanisms in which firmware can be distributed through IoT networks. Section VI considers the vulnerabilities firmware updates themselves may introduce into the transactive microgrid and identifies some management strategies to minimise the risk of corrupted updates while Section VII concludes the paper and highlights areas of future work for this study.

II. Background

Firmware update involves a sequence of events. The first four events are prerequisites for the second lot of events. The prerequisite events are as follows:

- First, the as-is device specifications (the current hardware and the current installed and executing firmware) must be known.
- Also, the need for the update must be clear, for example, a vulnerability in the current deployed firmware has been uncovered.
- Then, the new firmware requirements and associated tests specifications are developed.
- After which, the firmware is developed and tested against the test specifications.

This work is concerned about the firmware update events that follow the prerequisite events just described. The firmware update events are described next.

- Assuming the test results are sufficient, the firmware image is secured against tampering thus ensuring the integrity of the image. For example, a hash of the firmware image is calculated and signed using the manufacturer's private key. Since the scenario considered in this piece considers open-source solutions wherever possible, the image hash is signed but the firmware is not encrypted, keeping the image free to use for all.
- Then, the firmware image is distributed to the targeted node/s. The means by which this is done is considered elsewhere in this chapter.
- This is followed by the target node confirming the integrity of the received firmware image. One way to do this is by calculating a hash of the firmware, decrypting the signed hash using the manufacturer's public key, and comparing the decrypted value with the newly calculated hash. If the two correspond, then it is assumed that the received firmware image has not been altered and is safe to use in the update procedure.
- Then, the update is done at a time when the node is idle.
- Finally, regardless of the update result (successful or otherwise), the distribution party is informed of the outcome.

A. Considerations for Firmware Updating in the Transactive Microgrid

Some IoT devices, including the type considered in this document, may be in use for long period, even decades [7]. Part of device management includes firmware updating as a maintenance requirement [8]. Moran et al. [8] emphasises the necessity of a mechanism by which the firmware can be remotely updated by stating that it is critical to manage the life-cycle of the device (including the firmware), especially if the device is planned to be used for an extended period of

time, when the deployed device is located in an inaccessible area, or the cost to manually update the firmware is excessive. In addition, if an update mechanism is not available, then a device will function with the current vulnerabilities until such time as it is decommissioned [5]. Also, the ability to update firmware at a time after the device has already deployed makes it possible to do an initial deployment before all functionalities have been implemented. In addition, updates after deployment are a mechanism to address known firmware problems (including security vulnerabilities [9]), add security features [10], and make configuration changes. The changes possible are not limited to patches [11], but also complete firmware replacement. Additionally, optimisation updates [4] and functionality changes [8], [12] are possible using firmware updates.

B. Security requirements for firmware updating

Obtaining access to the firmware is a way to devise an attack, as it reveals valuable information about the libraries used, configuration settings, and functionality in general. Although reverse engineering the binary is not a trivial task, modern tools simplify this challenge [8].

Moran [8] advocates that the confidentiality of the firmware image should be protected so that the clear text binary is not exposed to unauthorized entities. Mtetwa [5] posits that firmware confidentiality is optional. However, firmware confidentiality is not an option if the firmware plain text is in the public domain such as the use case considered here. The issue of firmware confidentiality is therefore moot as far as this chapter is concerned.

While TLS channel is useful for confidentiality and integrity assurance, clear channel transmission has the benefit of transparency and security [7]. Bradley and Barrera [7] note that TLS alone is not provide complete protection against attacks specific to the update process. Update mechanisms using insecure communication channels remain an option for authentication and integrity assurance if out-of-band security is employed [7]. Digital signatures and certificates may be exchanged using insecure channels such as HTTP if the former are validated [7]. This can be seen in commercial off-the-shelf solutions such as the Apple TV protocol which uses HTTP channel to obtain the update and then uses remote attestation over HTTPS to authorise and validate the update.

Manual firmware update mechanisms that require the user to initiate the process are prone to human error [13]. Ideally, the end user should not participate it the firmware update process [14], [15]. Prakash et al. [16] states that the most effective update mechanism is a “silent” one; meaning the user is not involved in the update process. A benefit of unattended firmware updating mechanism is that it enables a scalable system [15]. Since the scenario considered here entails millions of devices, each potentially requiring one or more updates during its life, the manual firmware update approach is not appropriate and is not considered further.

C. Content distribution problem

According to Gkantsidis et al. [11], distributing firmware updates can be considered a content distribution problem, which has been well studied and several solutions have been proposed including multicasting, caching, and peer-to-peer networks.

Generally, to reduce on bandwidth and energy costs on the devices, it may be necessary to obtain firmware updates from multiple sources. As highlighted by Hernández-Ramos [14], this necessitates the establishment of a trust model between these sources and the device manufacturer. The purpose of this model is to ensure that only authorized and legitimate firmware is deployed on the relevant device. Key management mechanisms should be employed to address this issue. These mechanisms would embed the necessary cryptographic material into the IIoT device, thereby enabling the validation of new firmware images. Marchand et al. [3] identifies three firmware protection types, with these being secure update, attestation, and secure boot firmware.

III. Firmware Updating Models

Gupta and Van Oorschot [17] describes three firmware update models to consider. In all models, an appropriate update time must be determined. For example, if the IoT device is idle, the update could be implemented. This can be challenging because IoT devices may always be executing as is the case for a power meter [18], leaving no time to do an update.

A. Polling

The first is based on the end or client node polling the update source at regular intervals to determine if an update is available. This is called the pull model. Pull mode models of update are more suitable for higher powered IoT devices and gateways that are capable of supporting wider ranges of communication protocols as well as having higher energy resources owing to a continuous power source. In the polling model, the client node is responsible for the download, verification and installation of new firmware updates with the firmware server sending the image repository either via data structure or through a new image binary [1]. One of the main problems inhibiting the use of the pull model for IoT networks is the requirement for server registration and unique device identifiers. For large scale IoT networks, this becomes a costly exercise that inhibits the easy addition of new devices to the network and comes at a large resource cost to the firmware update server that needs to securely store and remember all the identifiers for the devices. Additionally, managing update scheduling is made difficult owing to the differing availabilities of devices and the real-time demands of some classes of devices always superceding the scheduling of other, lower priority devices. This could lead to some devices never being scheduled for an update until a point where the devices are completely de-synchronized from the available and supported firmware versions [1].

B. Pushing

The second is a push model in which the server initiates the firmware update and is more suitable for resource-constrained IoT devices with limited communication protocols. The responsibility of initiating updates in this model is pushed to the server and is only done when new patches are available. Like in polling mode, the end node would then download, verify and install the pushed patch update. To minimise the resource burden on nodes, server initiated updates are not conducted frequently unless patches are prioritised as in the case of zero-day or other highly critical vulnerability fixes [1]. The main problems identified with this model of updating is the lack of real-time update, which could increase exposure time of high risk vulnerabilities. Additionally, a significant sized update could pull sizable resource consumption on the firmware server in order to service the IoT network; a problem which scales in growth as the number of IoT devices in the network increases [1].

C. Negotiating

In the third model, negotiating or hybrid mode, flexibility is introduced towards updating different classes of IoT devices with differing requirements and availabilities. The update is negotiated once the server informs the client that a new image is available. In this model, the client determines if the update is required and may elect not to proceed with an update. Other negotiations may include instances where firmware is fully or partially updated. Partial updates are especially interesting in IoT environments as the devices where an update is being pushed may not have the necessary capabilities available for the full update but for mission criticality may still require a patch to be conducted for continued use. This third model aims to reduce the consumption of bandwidth and on-device processing resources by reducing the amount of code downloaded and installed and minimising the update time [1].

IV. Firmware Distribution Mechanisms for Transactive Microgrids

Significant network traffic is expected when multiple end nodes download firmware images from a single firmware update server [9]. In a client-server firmware distribution model, the network may not be capable of transporting the simultaneous request for firmware images from millions of IoT devices [5]. Updates to the client-server model need not have the firmware servers be physically hosted by the manufacturers. Instead, these can be cloud-based [5]. This approach has the advantage that the device supplier has close control over the devices but the disadvantage is that a centralised model is susceptible to a single point of failure [13].

Several previous works have looked at the implementation of decentralised and distributed firmware distribution mechanisms in lieu of the client-server model; most of which are primarily based on the use of blockchain. Decentralised firmware updating solutions are of great interest for application in the transactive microgrid because of the decentralised, distributed nature of the network. Within this study, six decentralised mechanism for IoT firmware updating were selected. A modified protocol by Kitchenham et al. [19] was followed using the following search terms:

"secure, "firmware", "update", "mechanism", "IoT".

As part of the manual exclusion criteria, only studies proposed within the last five years were considered, and consideration for the IoT device performance of the mechanism, as well as security against firmware updating attacks needed to be part of the analysis of the proposed solution.

The solution proposed by Lee and Lee [9] utilised the blockchain as a mechanism to verify firmware version, the authenticity of the version and distribute the specific binary to the IoT connected nodes within the blockchain networks. This was limited in application however owing to the higher energy, power and storage requirements required of a full blockchain node versus the light nodes that are typically used in transactive energy microgrids. This solution would be better suited only for Class 5 devices and not anything lighter weight.

Yohan and Lo [20] propose a Firmware-Over-The-Blockchain (FOTB) framework comprising of six main entities, most important of which are the active and passive blockchain nodes. The active blockchain node acts as a firmware broker and is owned by the vendor to create new firmware update contracts or update existing contract data while the passive nodes are at the end user side and receive and execute the firmware update contract from the active nodes. The framework combines a blockchain-based PUSH-PULL model between the active and passive nodes (assisted by the IoT gateway) with blockchain consensus being used to validate the firmware smart contract. Combined with the selection of Ethereum as their blockchain of choice, this assisted in reducing computational costs and providing protection against attacks such as firmware modification, impersonation, MiTM and Replay attacks.

Wijesundara et al. [21] propose an IOTA-based firmware updating framework using three main entities: an IoT vendor, IoT gateway and IoT end device. A MAM client is installed on the IoT vendor node to instantiate an IPFS connection between vendor and gateway nodes. this allows for the gateway to access and store information related to the IoT end device such as the type, MAC address, firmware version and firmware hash. IoT end devices then connect to the IoT gateway via MQTT servers. Two networks are used to distribute firmware through a combined PUSH-PULL model, IOTA Tangle and the IPFS network. Firmware is uploaded by the vendor and the Node.js portal calculates a SHA256 hash value which is then uploaded to the IPFS network and given an IPFS content ID for access through that associated network. Once publication to the IOTA Tangle is approved by the vendor, a MAM message is sent through a restricted channel containing all information relevant to the update. Gateways then cross-check compatibility against the IoT end device information they have available and push them to the end devices after running a SHA256-based hash verification on the firmware binaries. By switching from a blockchain to a DAG based

structure, the author's aimed to improve upon the scalability and latency issues from previous decentralised mechanisms as well as lowering financial and energy costs by eliminating the need for mining.

Dayaratne et al. [22] consider a firmware update management solution specifically targeted towards enabling and fostering trust in smart inverters within the transactive microgrid through the use of verifiable credentials and virtual power plants. For their proposed use case, a consortium blockchain is selected for security and decentralisation, given that freely open, public interaction with the network is not expected. This is to be expected in transactive microgrid deployments as participation would be limited to smaller communities and not open the trade market country wide or globally. A smart contract is deployed by the vendor into the blockchain to manage firmware update availability with evens being emitted only when new updates are available. With a public-private key pair, a public DID is generated for each smart inverter by the manufacturer and stored in its in-built wallet. The DID is used to subscribe the inverter to the smart contract events. When new updates are made available by the manufacturer, they are stored directly onto the blockchain or within an update server along with their associated verifiable credential that is signed using the manufacturer's private key. The smart contract verifies the signature on the update and once successful triggers an event to notify the availability of a new firmware version. Inverters subscribed to the contract events then check if the model for the update matches itself as well as whether the firmware version is newer than what is installed on the inverter. When both conditions are satisfied, the inverter saves the firmware's verified credentials to its wallet and uses its own verifiable credentials to securely access the download of the update. Once a successful update has occurred on the inverter, a new set of verifiable credentials are issued to the inverter with the updated firmware version via a user initiated request to the manufacturer.

Tsaur et al.'s [23] solution aims to ensure firmware integrity, malicious code resistance and distributed denial-of-service (DDoS) resistance within the IoT firmware update process. The proposed firmware updating system comprises of three main parts: network setup of the blockchain, file transmission and file downloading. A genesis node from the blockchain is utilised to deploy the smart contract and retrieve location data of other nodes within the network. When a manufacturer node initiates an update transaction, the update file is transmitted to other network nodes for verification through the blockchain. Once verified, the update file is stored in a distributed database with its associated address data stored in the blockchain ledger. This is done to reduce the storage requirements of the blockchain ledger as well as strengthening the network's robustness against DDoS attacks by eliminating the use of a single file server. When the IoT devices issue a query, the request is sent through the blockchain and the appropriate download file for the query is sent to complete the secure update.

Solomon et al [24] propose a blockchain based firmware updating framework designed with the resource-constrained nature of IoT devices in mind. To ensure lightweight operation, heavy computational tasks are off-loaded from the IoT device processing while a cost-effective smart contract is utilised for atomic payments for the exchanged software delivery. Device authorisation is implemented using a Ciphertext-Policy Attribute-Based Encryption (CP-ABE) for efficient, scalable key generation. The design of the framework considers three main types of malicious actors: a general malicious attacker, a malicious owner and a malicious manufacturer. As part of the threat model, software update notification attacks, confidentiality attacks, invalid update attacks, rollback attacks, non-delivery attacks, false update record attacks, and payment-free attacks are considered in the secure design. The framework utilises four main modules: key generation, software update notification, software cryptography, and update distribution and download. Using the CP-ABE, one key generation is needed for all devices satisfying the CP-ABE policy. This reduces the overhead required in comparison to unique key generation per IoT device. A notification message is issued on the blockchain by the manufacturer using a template for the software update information. The update decryption key is enrypted using CP-ABE and signed using ECDSA by the manufacturer prior to being deployed into the blockchain network. Manufacturer identity is verified by the smart contract via signature while comparison of the SHA-3 generated hashes are done to check for in-transit changes to the update. Owners, after paying the relevant smart contract fee, are then able to download the update from the cloud server via provided url and decrypt the update using the CP-ABE secret key, which is generated from attributes specific to the IoT device. Once the download is installed, the device notifies the blockchain with information about who participated in the update transaction-which is then used at a later stage by manufacturer and owners for auditing purposes.

Table I provides a summary of the firmware distribution mechanisms and compares them against three initial metrics for implementation consideration in the transactive microgrid: lightweight operation, broadband cost and security robustness. As maintaining availability of all hard-deadline operations is critical within the microgrid, assessment for lightweight operation of the firmware mechanism impacts the selection suitability for application within the microgrid. For this metric, consideration is given to the mechanism's the energy, computational, and storage requirements for the duration of the update against the resources available on the IoT devices being updated. The mechanism for updating the device firmware should not itself introduce latency or excessive energy and power consumption from the device in a manner that would compromise the smart microgrid operations. Lightweight design also expands the device class applicability of the updating mechanism, allowing for a wider reach of devices with the same mechanism regardless of the resources available on the device classes (as depicted in Fig. 1).

Broadband costs considers the various financial and time costs associated with the uploading and downloading of

TABLE I
COMPARISON OF SECURE FIRMWARE DISTRIBUTION MECHANISMS FOR TRANSACTIVE IOT MICROGRIDS

Solution	Lightweight Operation	Broadband Cost	Robustness against Security Attacks
Lee and Lee [9]	No	High	MiTM, Replay
Yohan and Lo [20]	Semi	Medium	Modification, Impersonation, MiTM, Replay
Wijesundara et al. [21]	Yes	Low	Firmware file authentication (SHA256)
Dayaratne et al. [22]	Yes	Low	Robust against falsification and enrolling inverters with outdated firmware
Tsaur et al. [23]	Yes	Low	Firmware integrity, malicious code resistance and distributed denial-of-service (DDoS) resistancee
Solomon et al. [24]	Yes	-	Robust against software update notification attacks (Blockchain), confidentiality attacks (CP-ABE), invalid update attacks (ECDSA, SHA3), rollback attacks (ECDSA, Smart Contract, Blockchain), non-delivery attacks (Smart contract, IPFS), false update record attacks (Blockchain), and payment-free attacks (ECDSA)

firmware updates to the IoT devices. This is also impacted by the communications channels used for updating as well as the signal strength and coverage in the areas in which the smart microgrid has been deployed, which would affect a devices' connection up and down time.

Lastly, the mechanisms are assessed for their robustness in protecting against a variety of firmware security attacks as part of the mechanism design.

In general, each of the analysed mechanisms were able to provide some form of security to the firmware updating process. In part, this is owing to the nature of blockchain; however coverage of the same attack vectors was not observed. Considering the environment in which the microgrid would be deployed and the sophistication of possible malicious attackers, as part of the microgrid threat analysis for firmware updating, a mechanism with sufficient robustness to cover the most high risk attacks would need to be considered.

Considering the requirements for lightweight operation, the solutions in [21], [23], [24], and [22] were developed with lightweight implementation in mind, making them more suitable candidates for a wider variety of IoT device classes. The solution by [22] specifically considers updating within the smart energy environment with a solution for inverter firmware updating, making it an attractive solution for implementation in the microgrid. Therefore, as part of future work, the implementation and performance tests of these mechanisms would need to be conducted on a variety of device classes to determine what would be the lowest device classes the distribution mechanism would be able to realistically service.

Consideration of the broadband costs yielded a similar result as the solutions by [21], [23], and [22] were seen as having lower costs when compared to the other selected mechanisms. Therefore, evaluation of the true costs when implemented as part of a physical evaluation test would also form part of an extended performance evaluation study in the future.

V. TOWARDS STANDARDISING FIRMWARE UPDATING

In recent years, work has been conducted by the Internet Engineering Steering Group (IESG) and the Software Updates for Internet of Things (SUIT) working group towards developing standardised firmware updating frameworks and protocols for IoT devices. One of the most widely regarded frameworks was proposed as an approved Request For Comment (RFC) document 9124 [15]. In this, the working group identifies and describes the elements that are required in a firmware manifest to ensure that IoT device firmware updates are secure. The manifest is defined as a machine-processible firmware image that includes metadata describing the firmware and information about the entity that created the firmware [15]. The manifest file includes, amongst others, information regarding when it was made, what other manifest files it needs, and how to check if the firmware image has been altered [25].

The RFC document covers a wide range of scenarios and the authors state that elements are to be selected based on the scenario under consideration. Table II presents a summary of elements that have been identified as Mandatory or Recommended within the RFC. (In Table II, "payload" refers to a firmware image that has been encrypted or compressed.) Standardisation of the firmware updating processes would aid in reducing the risk associated with patching and vulnerabilities associated with out-of-date device firmware. This would also aid in increasing the interoperability between IoT device

TABLE II
SUMMARY OF RFC 9124 MANIFEST INFORMATION ELEMENTS (MANDATORY AND RECOMMENDED)

Manifest Information Element	Description
Version ID of the Manifest Structure	Specifies the manifest data model used.
Monotonic Sequence Number	Prevents malicious firmware updates
Vendor ID	Aids in distinguishing between devices with same name but different vendors.
Class ID	Aids in determining if the device is compatible with the firmware update.
Payload Format	Required to decode the payload.
Processing Steps	How to decode a payload.
Storage Location	Specifies the location within a component.
Payload Digests	Determines payload/s authenticity.
Size	Size of the payload.
Manifest envelope element: Signature	Foundation for all manifest security.
Dependencies	Lists additional manifests required.
Encryption Wrapper	Points to the decryption key location.

classes as the fundamental process would remain the same regardless of device capability. Other works aiming towards the standardisation and regulation of IoT firmware updating including the United States of America's IoT Cybersecurity Improvement Act of 2020 [26], IETF RFC 9019 [27], and the ETSI EN 303 645: Cybersecurity Standard for Consumer IoT Devices [28].

VI. Vulnerability Management of Firmware Updates

While secure firmware updating can be one of the mechanisms used to keep microgrid devices up to date and protected against the latest vulnerabilities, the update images themselves can pose a high risk towards compromising the security and integrity of the microgrid network.

Bakhshi, Ghita and Kuzminykh [29] categorise IoT firmware vulnerabilities into eight main influencing factors (System Properties, Access Mechanisms, Component Re-use, Network Interfacing, Image Management, User Awareness, Regulatory Compliance and Adversarial Vectors) that encompass targeted common areas of exploitation. In the case of the transactive microgrid, some exploitation areas such as software and memory corruption, authentication, tainted data are of higher risk than others as they have safety and financial implications for end users should devices fail as a result of a firmware vulnerability [29].

Memory corruption vulnerabilities come as a result of programmatic errors such as buffer and integer overflows, pointer violations, missing bound checks, uncontrolled string formats among many other reasons. The memory constraints on IoT devices make these vulnerabilities occur more frequently in firmware because protections found in traditional PCs such as memory management units, are not included in the device design of lightweight IoT devices [30]. Software corruption vulnerabilities on the other hand occur as a result of a variety errors such as bugs, malicious code injections, backdoor testing openings such as hard-coded credentials and shared SSL certificates and outdated core components such as third party libraries [29]. Implementing good secure coding practices and frameworks into firmware development, such as DevSecOps, as well as introducing rigorous testing of firmware images during and post development could assist in reducing the risk of permutating these vulnerabilities into the transactive microgrid's IoT network. Other solutions such as software-defined networks and digital twins would also allow for a live deployment and monitoring of the firmware image to be conducted on an almost identical cyber system to pre-identify vulnerabilities and possibly problematic network behaviours before the firmware update is pushed onto the physical live network.

Authentication vulnerabilities are harder to solve in IoT networks owing to the scale and distribution of the network, the high level of automation without human intervention required, the decentralised nature of the network as well as the cyber-physical nature of IoT security vulnerabilities. As was identified with the software corruption vulnerability, in the development of IoT firmware, backdoors are often left in firmware during development by device vendors as part of testing strategies or as a mechanism for vendors to manage devices remotely. This has the result of bypassing authentication mechanisms installed on the device or implemented in the wider network, especially if these backdoors are unknown to the network owner. With the scale of IoT devices making implementation of traditional authentication solutions difficult and the often broadcast nature of IoT communications, the devices and their firmware become especially vulnerable to attacks such as false injection and brute force when attempting to gain control of the network [30]. In a critical application space such as the energy sector, this could lead to rolling blackouts should whole microgrid networks be compromised by unauthorised access via a vulnerable device. Physical insertion of a rogue device into the network could also serve as a mechanism to steal firmware intellectual property or compromise a previously secure firmware image if the identity of the rogue device cannot be authenticated with the firmware server and malicious actors are able to use it to gain access to the signed, secure firmware image. To counteract these vulnerabilities, default credentials should be changed prior to network deployment with strong password rules being applied. Proper access control mechanism and network joining protocols would also need to be implemented for the microgrid IoT network taking into consideration that ease of joining still needs to be maintained without compromising security.

For all firmware vulnerabilities, it is imperative that a recovery strategy is in place for the event that a firmware update fails [8]. This would form part of the disaster recovery policy and governance processes for the transactive microgrid and would include action such as roll-back to undo one or more recent updates [31]. The main aim besides containing the scale of the fallout would be to maintain as much network availability of the microgrid as possible with the remaining devices. This could even include limiting available energy trading areas or functionalities as predictive routing solutions identify the remaining physical infrastructure able to continue servicing the microgrid and maintaining grid stability.

VII. Conclusion

Designing an adequate firmware updating strategy for the IIoT requires there to be a high degree of autonomous operation due to the large device numbers seen within an IIoT network. Additionally, owing to the limited energy and processing resources available within IoT devices and the long lifespans of the networks (15+ years), updating solutions should be designed for low bandwidth and limited computational requirements. Updates need to be securely verifiable on deployment and prior to installation of the device to ensure that security vulnerabilities are not being introduced through firmware.

The study investigates and considers various firmware updating distribution mechanisms alongside manual firmware update approaches. Of the assessed strategies, the least suitable was found to be manual firmware updates owing to the large

number of devices present in IoT network distributions. The best suited strategies identified were decentralised, having highlighted two frameworks developed for lightweight IoT operation. However, each of these strategies have their associated limitations in practical use cases such as the smart microgrid network and would need further performance evaluation to be conducted as part of future work.

In addition to strategies for firmware distribution, this study identified vulnerabilities associated with the firmware images themselves. It was seen that many vulnerabilities could be mitigated through secure coding practices and careful design of authentication and access control measures. As part of future work, further investigation shall be done in identifying standards and frameworks related firmware vulnerabilities such as the IESG which has introduced an RFC in 2021 with a developed manifest of metadata to be included with the aim of identifying and maintaining the integrity of the firmware image. These elements in conjunction with other integrity preserving cybersecurity strategies could ensure faster vulnerability detection within firmware images prior to installation of a compromised image.

REFERENCES

[1] S. El Jaouhari and E. Bouvet, "Secure firmware over-the-air updates for iot: Survey, challenges, and discussions," *Internet of Things*, vol. 18, p. 100508, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660522000142

[2] L. P. I. Ledwaba, G. P. Hancke, S. J. Isaac, and H. S. Venter, "Smart microgrid energy market: Evaluating distributed ledger technologies for remote and constrained microgrid deployments," *Electronics*, vol. 10, no. 6, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/6/714

[3] A. Marchand, Y. Imine, H. Ouarnoughi, T. Tarridec, and A. Gallais, "Firmware integrity protection: A survey," *IEEE access*, vol. 11, p. 1, Jan 1 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10196361

[4] V. Nikic, D. Bortnik, M. Lukic, D. Danilovic, and I. Mezei, "Comparisons of firmware delta updates over the air using wlan and lpwan technologies." Piscataway: IEEE, Nov 15, 2022, pp. 1–4. [Online]. Available: https://ieeexplore.ieee.org/document/9983677

[5] N. S. Mtetwa, P. Tarwireyi, A. M. Abu-Mahfouz, and M. O. Adigun, "Secure firmware updates in the internet of things: A survey," in *2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*. IEEE, 2019, pp. 1–7.

[6] D. Kohout, T. Lieskovan, and P. Mlynek, "Smart metering cybersecurity—requirements, methodology, and testing," *Sensors (Basel, Switzerland)*, vol. 23, no. 8, p. 4043, Apr 17 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/37112383

[7] C. Bradley and D. Barrera, *Towards Characterizing IoT Software Update Practices*, ser. Lecture Notes in Computer Science. Switzerland: Springer Nature Switzerland, 2023, vol. 13877, pp. 406–422. [Online]. Available: https://library.biblioboard.com/viewer/135516e6-d039-11ed-8e0c-0a9b31268bf5

[8] B. Moran, H. Tschofenig, D. Brown, and M. Meriac, "A firmware update architecture for internet of things," 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9019

[9] B. Lee and J.-H. Lee, "Blockchain-based secure firmware update for embedded devices in an internet of things environment," *The Journal of supercomputing*, vol. 73, pp. 1152–1167, 2017.

[10] M. Bettayeb, Q. Nasir, and M. A. Talib, "Firmware update attacks and security for iot devices: Survey," in *Proceedings of the ArabWIC 6th Annual International Conference Research Track*, 2019, pp. 1–6.

[11] C. Gkantsidis, T. Karagiannis, and M. VojnoviC, "Planet scale software updates," ser. ACM Conferences. New York, NY, USA: ACM, Aug 11, 2006, pp. 423–434. [Online]. Available: https://search.proquest.com/docview/31426092

[12] E. Aras, S. Delbruel, F. Yang, W. Joosen, and D. Hughes, "Chimera: A low-power reconfigurable platform for internet of things," *ACM Transactions on Internet of Things*, vol. 2, no. 2, pp. 1–25, May 1 2021. [Online]. Available: https://lirias.kuleuven.be/handle/123456789/664476

[13] A. Yohan and N.-W. Lo, "An over-the-blockchain firmware update framework for iot devices." IEEE, Dec 2018, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/8625164

[14] J. L. Hernández-Ramos, G. Baldini, S. N. Matheu, and A. Skarmeta, "Updating iot devices: challenges and potential approaches," in *2020 Global Internet of Things Summit (GIoTS)*. IEEE, 2020, pp. 1–5.

[15] B. Moran, H. Tschofenig, and H. Birkholz, "Rfc 9124 a manifest information model for firmware updates in internet of things (iot) devices." 2022. [Online]. Available: https://datatracker.ietf.org/doc/rfc9124/

[16] V. Prakash, S. Xie, and D. Y. Huang, "Software update practices on smart home iot devices," Aug 30 2022. [Online]. Available: https://arxiv.org/abs/2208.14367

[17] H. Gupta and P. C. van Oorschot, "Onboarding and software update architecture for iot devices." Piscataway: IEEE, Aug 2019, pp. 1–11. [Online]. Available: https://ieeexplore.ieee.org/document/8949023

[18] J. Zhu, "Secure and automatic firmware update architecture for iot devices," 2018. [Online]. Available: https://datatracker.ietf.org/doc/draft-zhu-suit-automatic-fu-arch/00/.

[19] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering – a systematic literature review," *Information and Software Technology*, vol. 51, no. 1, pp. 7–15, 2009, special Section - Most Cited Articles in 2002 and Regular Research Papers. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584908001390

[20] A. Yohan and N.-W. Lo, "Fotb: a secure blockchain-based firmware update framework for iot environment," *International Journal of Information Security*, vol. 19, no. 3, pp. 257–278, 2020.

[21] W. Wijesundara, J.-S. Lee, D. Tith, E. Aloupogianni, H. Suzuki, and T. Obi, "Security-enhanced firmware management scheme for smart home iot devices using distributed ledger technologies," *International Journal of Information Security*, vol. 23, no. 3, pp. 1927–1937, 2024.

[22] T. Dayaratne, C. Rudolph, T. Shirley, S. Levi, and D. Shirley, "Fostering trust in smart inverters: A framework for firmware update management and tracking in vpp context," *IEEE Transactions on Smart Grid*, 2025.

[23] W.-J. Tsaur, J.-C. Chang, and C.-L. Chen, "A highly secure iot firmware update mechanism using blockchain," *Sensors*, vol. 22, no. 2, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/2/530

[24] G. Solomon, P. Zhang, R. Brooks, and Y. Liu, "A secure and cost-efficient blockchain facilitated iot software update framework," *IEEE Access*, vol. 11, pp. 44 879–44 894, 2023.

[25] J. L. Hernandez-Ramos, G. Baldini, S. N. Matheu, and A. Skarmeta, "Updating iot devices: challenges and potential approaches." Piscataway: IEEE, Jun 2020, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/9119514

[26] U. 116th Congress, "Iot cybersecurity improvement act of 2020," 2020. [Online]. Available: https://www.congress.gov/bill/116th-congress/house-bill/1668.

[27] B. Moran, H. Tschofenig, D. Brown, and M. Meriac, "Rfc9019: A firmware update architecture for internet of things," 2021. [Online]. Available: https://datatracker.ietf.org/doc/rfc9019/

[28] E. T. C. C. Security, "Etsi en 303 645: Cybersecurity standard for consumer iot devices," 2024. [Online]. Available: https://www.etsi.org/deliver/etsi_en/303600_303699/303645/03.01.03_60/en_303645v0

[29] T. Bakhshi, B. Ghita, and I. Kuzminykh, "A review of iot firmware vulnerabilities and auditing techniques," *Sensors*, vol. 24, no. 2, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/2/708

[30] I. Nadir, H. Mahmood, and G. Asadullah, "A taxonomy of iot firmware security and principal firmware analysis techniques," *International Journal of Critical Infrastructure Protection*, vol. 38, p. 100552, 2022.

[31] P. W. Osel and W. Gnsheimer, "Opendist - incremental software distribution," in *9th System Administration Conference (LISA 95)*, 1995 1995.