SHOID: A Secure Herd of IoT Devices Firmware Update Protocol

Frédéric Ruellé[a], Quentin Guellaën[b] and Arnaud Rosay[c]
STMicroelectronics, 11 Rue Pierre Félix Delarue, Le Mans, France

Keywords: Group of IoT Devices, Secure Firmware Update, Secure Group Communication, Device Management.

Abstract: The Internet Of Things (IoT) movement puts more and more objects on the field, which raises critical problems related to device management and security, especially for resource-constrained nodes. In this paper, we propose a method for the devices to create self-organized groups autonomously. We demonstrate how we can leverage this group concept to improve the robustness of a key device management procedure: the over-the-air firmware update. Experiments have been conducted with micro-controller based objects addressing a typical smart sensor use-case. Finally, ways of improvements arise and further study items are identified.

1 INTRODUCTION

Internet Of Things (IoT) is a major trend we can observe in almost all business areas. Bringing internet connectivity to all sorts of traditional devices is radically changing the way the internet is accessed and the way devices can be used. All these new connected objects can exchange data and commands with remote servers, also known as the cloud. As data exchanges are not always initiated by an explicit user request, security becomes a critical topic for the IoT. Hackers can access such devices remotely to steal data, use them as botnets to conduct other attacks like Distributed Denial Of Service attack, or misuse them in various manners. The Mirai botnet is an example of such attacks, and highlights the difficulty to thwart them without appropriate procedures (Kambourakis et al., 2017). Moreover, the number of attacks on IoT devices is increasing (Demeter et al., 2019). Therefore, IoT security is now a major topic that is also dealt with by laws (Kelly, 2020; European Commission, 2022). Professionals do insist on the critical importance of one procedure to secure IoT: the firmware update (Adams, 2021). So, many security companies and academics address this topic (ProvenRUN, 2022; K.Zandberg et al., 2019).

In addition to this security aspect, we observe the deployment of large fleets of resource-constrained devices in the area of smart lighting, smart building, smart cities, smart industries, and smart healthcare. Smart sensors have been defined some time ago (B.F. Spencer et al., 2004), but the domain evolves (Gervais-Ducouret, 2011). Nevertheless, the requirements for this class of devices are still the same: low production cost, battery powered device operating for years with low maintenance costs, limited data communication rates. These economical constraints induce strong technical constraints: limited memory and computing capacity, necessity to operate in low power modes with reduced data volumes. These requirements are key to deploy such sensors, and leverage all the benefits they bring: improved reactivity to real-time events, cost-efficient infrastructures monitoring and administration. Last but not least, due to the criticity of the envisioned applications, security is key, but must come at the lowest possible cost.

As the smart sensors of a same fleet share similar characteristics and run the same use-cases, we leverage these commonalities to improve the security of a group of devices by defining a synchronized secure firmware update procedure. Hence, we create a Secure Herd Of IoT Devices (SHOID), bringing extra security, but still meeting the economical and technical requirements listed above.

Our contribution applies to fleets of homogeneous objects sharing the same hardware, firmware and exchanging data with a same cloud infrastructure. We focus on the firmware distribution and validation by the devices. First, we review some existing techniques to create a secure group of IoT devices, and provide a secure firmware update service. Secondly, we propose an approach to group automatically and autonomously IoT nodes on the field, and make the

[a] https://orcid.org/0000-0002-4593-5032
[b] https://orcid.org/0000-0002-0372-072X
[c] https://orcid.org/0000-0001-5937-5331

Ruellé, F., Guellaën, Q. and Rosay, A.
SHOID: A Secure Herd of IoT Devices Firmware Update Protocol.
DOI: 10.5220/0011628000003405
In *Proceedings of the 9th International Conference on Information Systems Security and Privacy (ICISSP 2023)*, pages 373-380
ISBN: 978-989-758-624-8; ISSN: 2184-4356
Copyright © 2023 by SCITEPRESS – Science and Technology Publications, Lda. Under CC license (CC BY-NC-ND 4.0)

firmware update more secure. Thirdly, we describe the experimentation we conducted to validate these concepts.

2 RELATED WORK

Securing IoT nodes is a well-known challenge and many contributions propose some ways of doing it. The most common measures, fully relying on the device hardware, are: disk encryption, cryptoprocessor, signed boot and encrypted boot (Johnston et al., 2016). Our objective is to explore other solutions based on device cooperation instead of working at single device level.

2.1 Offload Security Measures to Edge Nodes

IoT nodes security can be addressed by delegating the security measures to some trusted network elements (Kuusijärvi et al., 2016). This approach concentrates the countermeasures in one place, and it becomes easy to upgrade these security functions. Nevertheless, this adds extra nodes in the system. Besides, the trusted Network Edge Device acts as an IoT gateway or supervises an IoT gateway. This requires significant computing power as the proposal is to use virtualized security functions. Deploying such powerful edge nodes does not meet some economical constraints.

Security measures can be offloaded to edge nodes with some Mobile Edge Computing nodes cooperating to share their defense resources and relax the budget constraints in each device (Li et al., 2021). But this requires extra components like the Cooperative Defense Orchestrator and complex scheduling algorithms. This approach, though lowering the cost per node, introduces specific elements and complexity.

2.2 Group Cryptographic Keys

Using groups is a common practice in the IoT as this enables convenient communication patterns like broadcast and multicast. Securing group communications, for instance to maintain the confidentiality of the exchanges, is key: encryption and hash algorithms bring confidentiality and integrity (Vinayaga Sundaram et al., 2015). To do so, cryptographic group keys are required.

A set of IoT nodes can agree on a group key (Gebremichael et al., 2018). This method requires a gateway collecting some secret numbers provided by the nodes joining the group. A benefit is that most of the complex operations are computed by the gateway, not the IoT devices. Nevertheless, this approach requires a dedicated functionality in a gateway and a secure communication channel between the gateway and the nodes. Besides, at device level, even when using some precomputed inputs from the gateway, the Elliptic Curve Point Multiplication is required. The Elliptic Curve Cryptography (ECC) induces performance overheads in terms of time, memory, and bandwidth penalty for authentication and encryption/decryption in Wireless Sensor Network applications (Shah et al., 2010).

An alternative to ECC is Advanced Encryption Standard (AES) Galois Counter Mode (GCM), which is suitable for IoT applications (Sung et al., 2018). AES is lighter than ECC (Mota et al., 2017). Another approach consists in using the block chain technology (Chen et al., 2021). Block chain is distributed and secure but involves additional infrastructure on top of the IoT network. Multicast techniques can also be used with symmetric keys, but specific actors like Network Multicast Managers can lead to a single point of failure issue as all elements do not play the same role (Carlier et al., 2018).

2.3 Managing Groups of Devices

Various methods exist to handle groups of objects from a communication and device management perspectives. Studies of secure IoT group communication state of the art are available, with a special focus on commercial smart lighting (Park et al., 2019).

One can group devices by assigning a specific role to one particular device (Park et al., 2018).

2.4 Firmware Update Methods

The secure firmware update of an IoT device is the number one requirement to enhance IoT security. A distributed firmware update architecture based on the blockchain provides the following benefits: decentralization, transparency, irreversibility (Choi and Lee, 2020). This solution implies the creation of various blockchain nodes with specific roles. Another blockchain based firmware update solution can use concepts like smart contracts, and gateways as block-chain clients (Tsai et al., 2020). The Ethereum blockchain platform can also be leveraged (Choi and Lee, 2020). These methods induce costs, and may require distribution incentives (Fukuda and Omote, 2021).

Studies depicting the landscape of firmware update solutions for IoT devices insist on the importance

of designing a comprehensive approach to ensure the integrity and an efficient management of the updating process (J.L.Hernández-Ramos et al., 2020). Another survey of the firmware update landscape provides a prototype based on a client-server approach and lists its challenges and the possible attacks (K.Zandberg et al., 2019). This architecture from the Software Updates for Internet of Things working group is deployed by major cloud providers like Microsoft or Amazon. Our contribution concerns this architecture.

2.5 Attacks

Many attacks against the firmware update procedure exist: tampered firmware, firmware replay, offline device attack, firmware mismatch, flash memory location mismatch, unexpected precursor image, reverse engineering, resource exhaustion (K.Zandberg et al., 2019).

The main silicon vendors provide device-level solutions to implement a secure firmware update. They usually offer software packages leveraging the hardware capabilities of the microcontrollers (NXP, 2020; STMicroelectronics, 2021b), so that low-end smart sensors cannot be cloned at a reasonable cost.

Nevertheless, these techniques are not sufficient when it comes to offline device attacks, denial of service attacks, or even tampered firmware injection. The dimension we leverage is the group, as we target fleets of devices sharing similar hardware and software capabilities, producing and processing similar sets of data. This group concept is the starting point of our contribution, enabling the definition of the secure synchronized firmware update procedure.

3 FIRMWARE UPDATE FOR DEVICE GROUPS

The IoT implies security challenges concerning the device communication and the device management. These challenges are particularly complex when it comes to resource-constrained devices such as smart sensors. We define a method for these devices to agree on a common cryptographic key, without using complex operations and specific infrastructures or dedicated provisioning mechanisms. This cryptographic key is the root of trust to define a protocol allowing the devices to group themselves autonomously, constituting self-organized groups of IoT nodes. The synchronized firmware update leverages these groups, enhancing the security of this key procedure in the context defined below.

3.1 System Security Perimeter

Our primary target is to increase the security level of a group of connected devices by making the firmware update procedure more secure. We focus on logical attacks exploiting the communication capabilities of the IoT nodes. Still, a certain level of resistance to physical attacks is required to prevent the leakage of critical security assets from the devices. We use the fortified solutions presented in 2.5 to secure the system assets against electronic board level attacks. We consider that the IoT nodes groups operate in multicast networking, like in a smart-building context (Li et al., 2013).

3.2 Group Key Creation

The devices to be grouped are homogeneous, implying that they come from the same vendor and run the same firmware version. This assumption can be relaxed by authorizing devices from various vendors, as long as they share enough commonalities evolving in a consistent and synchronized manner. Indeed, the group key creation mechanism relies on two assets: a secret cryptographic key, and a firmware image or a data image characterizing a set of devices. The secret cryptographic key can be an Original Equipment Manufacturer (OEM) key, provisioned for the needs of device-level secure boot and secure firmware update solutions (STMicroelectronics, 2021b). This key ensures the confidentiality, authenticity and integrity of the firmware images. It brings, by definition, a common secret shared by all similar devices. This secrecy is a primary security property while the application's firmware image is a source of diversity. By combining these two elements, we obtain an updatable common secret for the devices sharing these artifacts. Should an attacker purchase a same product model, still he has no guarantee that the product has been provisioned with the same OEM key and that it contains the exact same firmware image and data. So, we have both a strong secret, already provisioned for the purpose of the secure boot, and some variability, the firmware image, which brings a commonality between all the sensors of a same type. A secret key is derived from these elements. This key is common to a group of objects and evolves over the devices' lifetime. No further provisioning is required, but we need an algorithm processing these elements without requiring too much computing power. The use of Hash and Key Derivation Functions meets this objective.

All devices configured with the same OEM key and the same firmware image obtain the same secret key to protect their communications. It gives a ba-

sis to set up a group of IoT nodes communicating in a secure manner. The devices must have the same firmware image, but personalization is possible by keeping out of the hash perimeter some specific memory sections.

3.3 Group Creation Protocol

Our contribution groups devices of a same kind, so that they can cooperate to improve the security of the group. Devices sharing a same OEM key and a same firmware image compute a common secret key. Besides, we use symmetric cryptography for performances reasons. In this area, Authenticated Encryption with Associated Data algorithms combine authentication and encryption to ensure source authentication, as well as data integrity, in addition to confidentiality. These techniques are interesting (Bellare and Namprempre, 2000), and AES GCM is an efficient solution for Cortex-M based devices (Driscoll, 2018; Koteshwara et al., 2019; Sovyn et al., 2020). It generates two outputs: an encrypted version of the data it processes, and a tag allowing to authenticate this data. These properties ensure the confidentiality, authenticity and integrity of the messages exchanged between the IoT nodes sharing the same secret key.

This secret key is used to select the IoT devices that can join a same group, without defining this group a priori, and without any node playing a specific role. Devices deployed on the field create groups autonomously. This self organizing network of IoT nodes requires a common channel that devices use to discover each other. This channel can be a multicast channel to restrict the considered perimeter to a given area. The way to define it depends on the deployment model selected for the system. Several groups can use the same common channel if they have different secret keys. A common channel can be defined for a building or a district. Let's take the example of a smart building: all devices deployed in the building share the same common channel. But, the connected thermostats have a secret key while the smart light bulbs share another secret key. Hence, autonomously, the devices group themselves, and each time a new device is commissioned it joins the proper group automatically, without any supervisor overseeing the system.

The protocol to do so is a set of messages made of two parts: a non-encrypted header (plain text) providing all the information required to use the payload, and an encrypted and authenticated payload. AES GCM may also use the clear header as Additional Authenticated Data, so that the entire message is authenticated. The encrypted payload carries the sensitive information: the identifiers of the sender and receiver of the message, a sequence number against replay attacks, the value of the message and a signature to authenticate the sender. On message reception, the device uses the clear header and the secret key to perform the AES GCM decryption and the authentication tag verification. When processing the payload, another level of verification is performed thanks to the signature, to make sure the sender has not been impersonated.

The group creation and update relies on two messages. On one hand, a HELLO message is sent by a newcomer on the common channel. This message indicates that the device is active and ready to receive notifications from group members. On the other hand, the members of the group send a HELLO RESPONSE message to the newcomer so he can discover them. As these messages are encrypted, only the devices with the appropriate secret key can understand them. This is how the group is built and maintained over time. These messages carry information about their sender: its identifier and its personal communication channel. Only the node owning the personal channel posts on it, and the signature field of the message ensures this. All other devices of the group listen to this channel. This system uses a common channel to create and maintain the group, and a device-specific channel, so that nodes can send specific information to the listeners.

A device that fails to perform its firmware update while the rest of the group succeeds to do so is, by construction, kicked out of the group. This description focuses on the principles to create the group and skips the messages required for dealing with situations where a device becomes unreachable, out of order, or replaced.

3.4 Synchronized Firmware Update

The group of devices is the foundation for improving the overall security of the fleet. The firmware update is a critical operation for managing IoT nodes and maintaining their security mechanisms up to date. But, this feature offers an attack surface for DoS attacks or malicious code injection attacks. In order to address these issues, we define one group procedure: the synchronized firmware update. This contribution focuses on this sole procedure, but additional procedures can be used to maintain the group's validity over time.

Concept. The synchronized firmware update consists in two steps: first making sure all devices of the group have received the same update request, then electing a device that runs the update first to confirm

its validity. If the devices receiving the firmware update request do not receive the update response messages from the other group members then the request is discarded. The entire fleet must be updated at once for the update request to be considered trustworthy. As the group key is based on the firmware image, a partial update of the fleet would result in breaking the group. Some devices can be unreachable, due to sleep mode, out of order state, or replacement. This might create a security breach, so more refined heuristics are required at deployment stage. To sum up, this system prevents a hacker from targeting one single device: he needs to find out the group members, and attack them all in a synchronized manner. Moreover, the system relies on group decisions, so hijacking one device is not sufficient to control the decisions. To avoid a misuse of the group decisions, it is important to limit the number of new devices joining the group simultaneously. This risk is managed by enabling device-level protections to prevent devices cloning or counterfeiting.

When this synchronized request is received, a device is elected to run the firmware update first. This device plays the role of a scout, and must confirm the update success before the group carries on with the update. This lowers the risks of incorrect update, without involving any supervision. No operator is needed to select a test device before propagating the firmware update to the entire fleet. All devices are addressed at once, and they handle the update themselves. For an attacker, it is very complicated to guess who is going to be chosen as the scout. While the test device performs its update, the other devices put their own update procedures on hold, but they can still receive commands and run their use-cases. After a successful update, the test device confirms the validity of the new firmware to the group. If this confirmation is not received within a guard period, then the other devices discard the update request, blacklist the candidate firmware version, notify the cloud of this erroneous update and of the faulty test device that might run a malicious code, enter a silent period where all subsequent firmware update requests are ignored. Hence, the rest of the fleet is not updated, the server is notified and can launch the appropriate countermeasures. If the confirmation is received within the guard period, then all the other devices of the group proceed with the firmware update.

The procedure starts when an UPDATE REQUEST message is received by each device, and specifies the new firmware image to be installed. Each node runs the algorithm 1 on top of the generic algorithm 2. Once the firmware update has been completed successfully, the test device has a new firmware image in place, so it generates a new group key. Therefore, this device must recompute the previous group key to send the confirmation messages before the entire herd moves to the new firmware image and its associated group key.

Threats Towards the Group Itself. An attacker might try to break the group itself to circumvent the synchronized firmware update procedure. Therefore, the communications are protected (confidentiality, authenticity and integrity) and replay attacks are prevented by the use of messages sequence numbers. Last but not least, device level protections and monitoring procedures are additional layers of security preventing the hacker from hijacking a device, and using it to dismantle the group.

Synchronized Firmware Update Benefits. A first benefit concerns the maintenance of the fleet. The fleet owner does not need to maintain explicitly a list of his groups of devices. He can send a general update request to deploy a new firmware version. The devices are self-organized per groups and each group elects a test device. This test device makes sure the new firmware is correct (self-tests) before authorizing the update of the peers. If one device cannot update for any reason, then it is automatically removed from the group, as it cannot compute the new group key, and this failure is reported to the server. A second benefit concerns the security of the herd, as an attacker needs to attack all devices of a group at once. First, he needs to determine the group members. Then, he must update the binary image he is using as soon as a firmware version is blacklisted, and forge a valid firmware image able to send the UPDATE CONFIRMATION message. Infecting the other devices of the group requires knowing the group, computing the proper group key and forging the proper acknowledgement message. Hence, the synchronized firmware update procedure raises the bar for attackers: code injection and DoS attacks are thwarted. Last but not least, it brings an automatic device management capability.

4 EXPERIMENTATION

IoT Node. We conducted experiments with microcontrollers to verify the feasibility of the system and its efficiency. We used STMicroelectronics' STM32L4 Discovery kit IoT node B-L475E-IOT01A (STMicroelectronics, 2021c) because it is representative of a mid-range connected sensor. It provides an IEEE 802.11 (Wi-Fi) connectivity to establish a link

Algorithm 1: Synchronized Firmware Update.

```
The device is part of the group G
The Firmware Update Request R carries
  firmware version R.version = N + 1
Function processUpdateRequest(R:
  request):
    version ← R.version
    number ← randomnumber
    create UPDATE RESPONSE message
      with version and number
    start guard timer TG for group responses
    for device in G do
        send UPDATE RESPONSE to device
    end
    Wait for all responses to be received
    if all responses received then
        Stop guard timer TG
        if my random number is the highest
          one in G then
            memorize that an update is in
              progress with
              UpdateInProgress flag
            perform the firmware update
              procedure
            Reset()
        else
            Start guard timer TC for test
              device confirmation
            Wait for test device to send the
              UPDATE CONFIRMATION
              message
            if UPDATE CONFIRMATION
              received then
                stop TC
                perform the firmware update
                  procedure
                Reset()
            else
                TC expires
                Abort the synchronized
                  firmware update procedure
                Notify the server that a
                  synchronized firmware
                  update procedure failed
    else
        Cancel the update procedure when
          TG expires
        Notify the server that an abnormal
          update request has been received
          and aborted
return
```

Algorithm 2: Device generic loop.

```
The device runs OEM Firmware N
The device has the OEM firmware key K_OEM
Function Main():
    initialization: compute the group key
      K_cur and establish the group G if
      needed
    if UpdateInProgress flag is set then
        validate the new firmware image
        reset the UpdateInProgress flag
        compute the former group key K_prev
        create UPDATE CONFIRMATION
          message using K_prev
        for device in G do
            send UPDATE CONFIRMATION
              to device
        end
    end
    while True do
        do use-case processing
        get message M
        if M.type == UPDATE REQUEST
          then
            processUpdateRequest(M)
        end
    end
return
```

with a remote cloud server. Besides, it also provides a Media Access Control address we can use as the device identifier within the group. Many sensors are available on this board so we can develop a realistic use-case: a connected thermostat. On software side, we implement our system on top of two STM32 packages (STMicroelectronics, 2021a; STMicroelectronics, 2021b).

Results. The microcontroller takes 483 ms to compute the group key, with Secure Hash Algorithm 256 bits, when running at 80 MHz with a 251 kB firmware image in FLASH. With a group of n devices with $n = 4$, the system generates 12 MQTT messages for a total of around 1500 bytes exchanged over the network. This is $n \times (n-1)$ messages, so the number of messages goes in $\mathcal{O}(n^2)$, showing that group size matters. The duration to establish the HTTP connection to the update server, then download the 251 kB firmware image and deem it is invalid is a few tens of seconds. During this period, a Wi-Fi IoT Node is in a TX state. According to a study of IoT power consumption, an IoT node operating in this TX state drains a current of 0.38 A from a 3.3 V power supply (Guegan and Anne-Cécile, 2019). Therefore, skip-

ping these useless steps induces significant energy savings. Besides, the update server processes only 1 instead of n connections. For a fleet of 4 devices, almost 750 kB of useless data transfer is saved. These energy and bandwidth savings evolve in $\mathcal{O}(n)$, so the result is worth the messaging overhead. The system adds 10 kB of code in FLASH, for a reference firmware size of 237 kB (sensor management code, MQTT, TLS and device drivers libraries) and no extra RAM budget is required.

5 CONCLUSION

In this paper, we studied a way to group IoT devices to strengthen the security of the firmware update procedure. The method establishes a common cryptographic key, and uses it to group IoT nodes. This technique requires no compute-intensive processing and no specific provisioning of the devices. The secure group concept is used to create a synchronized firmware update procedure. Besides, this system requires very few server interactions as being fully based on a device to device approach. This property brings additional autonomy and robustness benefits.

In future work, a more efficient device to device communication could be exploited to improve power consumption and server independence, by using Long Range Wide Area Network (LoRaWAN) or Bluetooth Mesh networking. Last but not least, we may refine the heuristics to improve the system efficiency: guard periods, unreachable devices management, power saving impact on procedures.

ACKNOWLEDGEMENTS

The authors would like to thank Julien Montmasson (ST Microelectronics) and Géraud Plagne (ST Microelectronics) for their valuable comments and suggestions to improve this contribution. The authors would also like to thank Eloise Cheval (ST Microelectronics) for her thorough reviews of this paper.

REFERENCES

Adams, E. (2021). Protect your customers with a secure iot updating process. https://blog.securityinnovation.com/secure-iot-updating-process. Last checked on Oct 05, 2022.

Bellare, M. and Namprempre, C. (2000). Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Okamoto, T., editor, *Advances in Cryptology — ASIACRYPT 2000*, pages 531–545, Berlin, Heidelberg. Springer Berlin Heidelberg.

B.F. Spencer, J., Ruiz-Sandoval, M. E., and Kurata, N. (2004). Smart sensing technology: Opportunities and challenges. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.3329&rep=rep1&type=pdf. Last checked on Oct 05, 2022.

Carlier, M., Steenhaut, K., and Braeken, A. (2018). Symmetric-key based security for multicast communication in wireless sensor networks. In *2018 4th International Conference on Cloud Computing Technologies and Applications (Cloudtech)*.

Chen, C.-M., Deng, X., Gan, W., Chen, J., and Islam, S. K. H. (2021). A secure blockchain-based group key agreement protocol for iot. *The Journal of Supercomputing*, 77(8):9046–9068.

Choi, S. and Lee, J.-H. (2020). Blockchain-based distributed firmware update architecture for iot devices. *IEEE Access*, 8:37518–37525.

Demeter, D., Preuss, M., and Yaroslav, S. (2019). Iot: a malware story. https://securelist.com/iot-a-malware-story/94451/. Last checked on Oct 05, 2022.

Driscoll, K. (2018). Lightweight crypto for lightweight unmanned arial systems. In *2018 Integrated Communications, Navigation, Surveillance Conference (ICNS)*, pages 1–15.

European Commission (2022). Cyber resilience act. https://digital-strategy.ec.europa.eu/en/library/cyber-resilience-act. Last checked on Oct 04, 2022.

Fukuda, T. and Omote, K. (2021). Efficient blockchain-based iot firmware update considering distribution incentives. In *2021 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–8.

Gebremichael, T., Jennehag, U., and Gidlund, M. (2018). Lightweight iot group key establishment scheme using one-way accumulator. In *2018 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–7.

Gervais-Ducouret, S. (2011). Next smart sensors generation. In *2011 IEEE Sensors Applications Symposium*, pages 193–196.

Guegan, L. and Anne-Cécile, O. (2019). Estimating the end-to-end energy consumption of low-bandwidth.

J.L.Hernández-Ramos, G.Baldini, S.N.Matheu, and A.Skarmeta (2020). Updating iot devices: challenges and potential approaches. In *2020 Global Internet of Things Summit (GIoTS)*.

Johnston, S. J., Scott, M., and Cox, S. J. (2016). Recommendations for securing internet of things devices using commodity hardware. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pages 307–310.

Kambourakis, G., Kolias, C., and Stavrou, A. (2017). The mirai botnet and the iot zombie armies. In *MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM)*, pages 267–272. IEEE.

Kelly, R. (2020). H.r.1668 - iot cybersecurity improvement act of 2020. https://www.congress.gov/bill/

116th-congress/house-bill/1668. Last checked on Oct 05, 2022.

Koteshwara, S., Das, A., and Parhi, K. K. (2019). Architecture optimization and performance comparison of nonce-misuse-resistant authenticated encryption algorithms. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(5):1053–1066.

Kuusijärvi, J., Savola, R., Savolainen, P., and Evesti, A. (2016). Mitigating iot security threats with a trusted network element. In *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 260–265.

K.Zandberg, K.Schleiser, F.Acosta, H.Tschofenig, and E.Baccelli (2019). Secure firmware updates for constrained iot devices using open standards: A reality check.

Li, D., Aung, A., Sampalli, S., Williams, J., and Sanchez, A. (2013). Privacy preservation scheme for multicast communications in smart buildings of the smart grid. *Smart Grid and Renewable Energy*, 4(4):313–324.

Li, H., Yang, C., Wang, L., Ansari, N., Tang, D., Huang, X., Xu, Z., and Hu, D. (2021). A cooperative defense framework against application-level ddos attacks on mobile edge computing services. *IEEE Transactions on Mobile Computing*. Publisher Copyright: IEEE.

Mota, A. V., Azam, S., Shanmugam, B., Yeo, K. C., and Kannoorpatti, K. (2017). Comparative analysis of different techniques of encryption for secured data transmission. In *2017 IEEE International Conference on Power Control Signals and Instrumentation Engineering (ICPCSI)*.

NXP (2020). Lpc55s00 security solutions for iot. https://www.nxp.com/docs/en/application-note/AN12278.pdf. Last checked on Oct 05, 2022.

Park, E., Kim, K., and Kim, N. (2018). Group management scheme to support robustness service in iot environment. volume 118, pages 1183–1194.

Park, J., Jung, M., and Rathgeb, E. P. (2019). Survey for secure iot group communication. In *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 1026–1031.

ProvenRUN (2022). Firmware update is a must, but needs also to be secured. https://provenrun.com/products/provenfota/. Last checked on Oct 05, 2022.

Shah, P. G., Huang, X., and Sharma, D. (2010). Analytical study of implementation issues of elliptical curve cryptography for wireless sensor networks. In *2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*, pages 589–592.

Sovyn, Y., Khoma, V., and Podpora, M. (2020). Comparison of three cpu-core families for iot applications in terms of security and performance of aes-gcm. *IEEE Internet of Things Journal*, 7(1):339–348.

STMicroelectronics (2021a). Generic iot cloud software expansion for stm32cube. https://www.st.com/en/embedded-software/x-cube-cld-gen.html. Last checked on Sep 05, 2021.

STMicroelectronics (2021b). Secure boot & secure firmware update software expansion for stm32cube. https://www.st.com/en/embedded-software/x-cube-sbsfu.html. Last checked on Sep 05, 2021.

STMicroelectronics (2021c). STM32L4 discovery kit iot node, low-power wireless, BLE, NFC, SubGHz, Wi-Fi. https://www.st.com/en/evaluation-tools/b-l475e-iot01a.html. Last checked on Sep 05, 2021.

Sung, B.-Y., Kim, K.-B., and Shin, K.-W. (2018). An aes-gcm authenticated encryption crypto-core for iot security. In *2018 International Conference on Electronics, Information, and Communication (ICEIC)*, pages 1–3.

Tsai, M.-H., Hsu, Y.-C., and Lo, N.-W. (2020). An efficient blockchain-based firmware update framework for iot environment. In *2020 15th Asia Joint Conference on Information Security (AsiaJCIS)*, pages 121–127.

Vinayaga Sundaram, B., M., R., M., P., and J., V. S. (2015). Encryption and hash based security in internet of things. In *2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*, pages 1–6.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS