Towards Better Availability and Accountability for IoT Updates by means of a Blockchain

Aymen Boudguiga, Nabil Bouzerna, Louis Granboulan, Alexis Olivereau, Flavien Quesnel, Anthony Roger, Renaud Sirdey

Towards Better Availability and Accountability for IoT Updates by means of a Blockchain

Aymen Boudguiga*, Nabil Bouzerna*, Louis Granboulan†, Alexis Olivereau‡,
Flavien Quesnel*, Anthony Roger* and Renaud Sirdey‡*
*IRT SystemX, 8 avenue de la Vauve 91120–Palaiseau, *firstName.lastName@irt-systemx.fr*
†Airbus Group Innovations, *louis.granboulan@airbus.com*
‡CEA–LIST, 91191 Gif-sur-Yvette, *firstName.lastName@cea.fr*

***Abstract*—Building the Internet of Things requires deploying a huge number of devices with full or limited connectivity to the Internet. Given that these devices are exposed to attackers and generally not secured-by-design, it is essential to be able to update them, to patch their vulnerabilities and to prevent hackers from enrolling them into botnets. Ideally, the update infrastructure should implement the CIA triad properties, i.e., confidentiality, integrity and availability. In this work, we investigate how the use of a blockchain infrastructure can meet these requirements, with a focus on availability.**

***Index Terms*—Blockchain, IoT, Software updates, Availability, Accountability, Innocuousness**

I. Introduction

The Internet of Things (IoT) is today a paradigm that is changing our lives in many ways with respect to how we work, travel, entertain ourselves or communicate. Novel forms of interactions with human users, distributed intelligence, pervasiveness, new topologies, devices and communication technologies shape this evolution. From the non-expert viewpoint, the digitalization of hitherto human-centric scenarios and the devices by which this digitalization is carried out are its most visible aspects. Practically, it results in a profusion of new computerized objects and appliances. It is expected that by 2030, the number of connected devices will reach 30 billions and they will exchange hundreds of zettabytes of data.

Individually, it is well-known that each IoT device exhibits specific vulnerabilities to cyberattacks due to multiple factors including longevity, lack of physical protection, hardware shortcomings or stripped down Human Machine Interface. With such a mass deployment, the vulnerability property therefore calls for universal software update infrastructures enabling IoT product manufacturers / integrators to remotely maintain software-based equipment. Yet, even the best conceived software update platforms can fall short if they are deliberately targeted as part of a combined attack scenario.

In this work, we investigate the possible use of a blockchain infrastructure to provide software updates to several IoT objects belonging to different manufacturers. As a blockchain relies principally on a peer-to-peer network, we thought intuitively that it may serve as a distributed database for storing and sharing software updates between IoT objects.

In this paper, we show how IoT manufacturers can benefit from the use of a blockchain to ensure updates *availability* and *innocuousness* for IoT objects. Availability and also integrity result from the persistence property of the blockchain. That is, once an update is added to the blockchain as part of a valid block, it becomes impossible to erase it. As such, we defeat malicious entities that prevent software updates from being distributed, in order to benefit from current software vulnerabilities. That is, we ensure that updates will be always available for their intended devices. Moreover, the blockchain infrastructure may be assumed to be much more resilient to availability threats e.g., impersonations or Denial of Service (DoS), than the manufacturer's own infrastructure.

In addition, we propose to rely on some nodes in the blockchain to validate an update innocuousness before its transmission to end devices. Innocuousness checking nodes will not only check the integrity of the updates by verifying manufacturers signatures but they will also validate the innocuousness of the received software by checking it for bugs, analyzing its vulnerabilities and testing its resistance to a set of known attacks. As such, devices will only download an update that has been approved by some innocuousness checking nodes. These nodes can belong, for example, to national cybersecurity agencies. That is, we protect legitimate devices from downloading insecure updates.

The remainder of this paper is organized as follows. Section II presents the targeted IoT architecture. Section III depicts two mechanisms for software update with a blockchain. Section IV concludes the paper and gives some future works.

II. Targeted IoT Architecture

Tschofenig and Arkko [1] present smart objects as devices with size, memory, computation or energy constraints. These devices are used in our daily lives to return interesting information about our environment. For example, in smart homes, smart objects may serve to return temperature levels or our energy consumption rates. Smart objects can connect to the Internet to form an Internet of Things (IoT) architecture. Tschofenig et al., [2] distinguish three models of IoT communication:

- *Device-to-Device communication* concerns smart objects that exchange information in a peer-to-peer manner without accessing the Internet. For example, in a smart home, a wireless communication can be established between a light bulb and a light switch that do not come necessarily from the same manufacturer.
- *Device-to-Cloud communication* refers to use cases where a smart object interacts with a service provider in the Internet. For example, a temperature sensor can transmit real-time information about a smart house temperature to an energy service provider. The latter will define the user

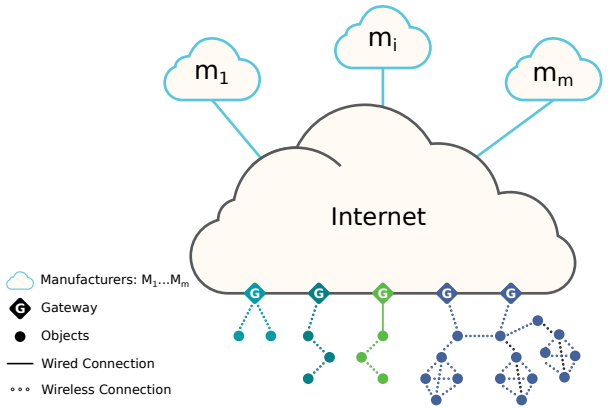

Fig. 1. Abstract IoT architecture

energy consumption profile and output advice to reduce his energy bill.

- *Device-to-Gateway communication* either provides Device-to-Cloud communication via a routing gateway or relies on an applicative gateway to gather device data. A routing gateway can be for example a smartphone that relays a user health data collected by a smartwatch to a service provider database. Meanwhile, an applicative gateway can gather, for example, temperature data from different sensors and synthesizes the results for the end user.

Figure 1 describes an abstract IoT architecture. First, we define a set of device *manufacturers* $\mathsf{M} = \{\mathsf{m}_1, \ldots, \mathsf{m}_\mathsf{m}\}$. m_i can be a smart meter, a car or even a smart fridge manufacturer. It is worth noting that we have a $1 : \mathsf{n}$ relationship between a manufacturer and its own devices.

Second, we define $\mathsf{O}_\mathsf{i} = \{\mathsf{o}_{\mathsf{i}1}, \ldots, \mathsf{o}_{\mathsf{in}}\}$ as the the set of smart *objects* manufactured by m_i. That is, each $\mathsf{m}_\mathsf{i} \in \mathsf{M}$ is managing a set of devices O_i. For simplicity sake, we suppose that each manufacturer m_i is in charge of n distinct devices. As such, the cardinality of $\mathsf{O}_{\mathsf{i},\forall \mathsf{i} \in [1,\mathsf{m}]}$ equals n.

As presented in Figure 1, a device can connect to the Internet, and so to its manufacturer cloud, via a *gateway* G, directly or by hoping through other devices until getting access to the gateway G. That is, a device can participate into a peer-to-peer communication to relay its peers' traffic to a certain gateway. A gateway can simply be a WiFi Access Point in a smart home or a Road Side Unit in an Intelligent Transportation Structure (ITS) architecture. Note that devices forming a peer-to-peer network do not belong necessarily to the same manufacturer. For example, different brands of cars communicate directly in an ITS context.

III. Blockchain and Software Updates

In this section, we describe how IoT manufacturers can benefit from the use of a blockchain to provide devices updates availability and innocuousness. For simplicity reasons, we will talk only about software updates. However, our presented solutions can be adapted to applications distribution, to network settings sharing or to new configuration profiles dispatching.

We first define the considered attacker model in section III-A. Then, we depict keys initialization mechanism in section III-B. Keys are compulsory for updates signing and encryption, and for blockchain operations. In section III-C, we introduce our first contribution. We show how we can benefit from a blockchain as a distributed database to share updates between several devices. That is, we show how a blockchain can provide updates *availability*. Then, we go a step further by adding a set of *innocuousness* checking nodes in the blockchain (section III-D). These nodes are in charge of validating manufacturers updates before their transmissions to the end devices. That is, these nodes will not only check the integrity of the updates by verifying manufacturers signatures but they will also validate the innocuousness of the received software by checking it for bugs, analyzing its vulnerabilities and testing its resistance to a set of known attacks. The innocuousness checking nodes can be, for example, national cybersecurity agencies and certifying cybersecurity companies. An object will not download an update until it has been approved by a set of innocuousness checking nodes. Finally, section III-E presents an update acknowledgement protocol. Acknowledgement is important as it creates a *history* of installed updates.

A. Attacker Model

In this work, we consider a Dolev and Yao attacker model [3]. That is, the attacker is able to read, send and drop a transaction addressed to the blockchain, or any network packet. Of course, she can be passive by connecting to the network and eavesdropping all exchanged messages. Or, she can be active by injecting, replaying or filtering exchanged information.

Our attacker targets devices, their manufacturers, the update blockchain or even the network:

- Attacking the network: an attacker can try to isolate a device or its manufacturer to prevent it from sending a transaction to the blockchain. Here, by isolation, we refer to classical network attacks where the traffic of the attacked node is filtered, or the link between the targeted node and its network is simply cut. As such, a network attacker can prevent a device from sending its acknowledgment transaction of section III-E. Network isolation attacks can be simply avoided by ensuring that each device or manufacturer has in its routing table redundant paths to the core network.
- Attacking the blockchain: as the update transactions T or the acknowledgment transactions A are signed by manufacturers and devices respectively, the attacker will not be able to impersonate a manufacturer or a device unless she gets their respective signing keys. However, she can try to prevent a legitimate transaction from appearing into a valid blockchain block, which is equivalent to the double spending problem in bitcoin [4]. By construction, the attacker will have to control more than half of the blockchain nodes to prevent a transaction from appearing in a valid block in the blockchain which is presumably hard [4]. Moreover, manufacturers can check that their updates correctly appear in the blockchain; if this is not the case, it can mean that someone is attacking the blockchain.
- Attacking a manufacturer: as all update transactions T are signed with manufacturers private keys, the attacker will not be able to impersonate a legitimate manufacturer.

However, a passive attacker can recover all updates binaries. As such, manufacturers will have to encrypt their updates binaries when their confidentiality e.g., intellectual property, is a concern.

- Attacking a device: as an attacker can hack into a device where keys are stored in a HSM, she is able to recover all the updates, even the encrypted ones. Indeed, she has simply to dump the memory of the hacked device to get the last software update. In this work, we do not address the problem of confidentiality within hacked devices.

B. Keys Initialization

We consider a *manufacturer* (m_i) who sells some *devices* (O_i) and who, for some good and legitimate reasons, wishes to perform software updates on these devices once fielded. Let us assume that the manufacturer has its own master public/secret keys pair, say pk^{m_i} and sk^{m_i}, and that it also generates such a pair for each device o_{ik}, say $pk^{o_{ik}}$ and $sk^{o_{ik}}$. Before shipping device o_{ik}, the manufacturer somehow "burns" pk^{m_i}, $sk^{o_{ik}}$ and $cert^{o_{ik}}$ into it[1]. $cert^{o_{ik}}$ is the tuple $(pk^{o_{ik}}, sign^{m_i}(sk^{m_i}, pk^{o_{ik}}))$ where $sign^{m_i}(sk^{m_i}, pk^{o_{ik}})$ is the signature of $pk^{o_{ik}}$ with the private key of m_i, namely sk^{m_i}. Today, it is common use to rely on a Hardware Security Module (HSM) or on a tamper resistant memory for secure keys storage. So, it is fair to assume that m_i is storing its pk^{m_i}, $sk^{o_{ik}}$ and $cert^{o_{ik}}$ on its devices HSMs.

C. Blockchain and Updates Availability

In order, to generate an update U for device o_{ik}, m_i proceeds in a utterly classical fashion by signing U by means of sk^{m_i} and then encrypting U and the signature under $pk^{o_{ik}}$. Upon receiving (or retrieving) an update U, the device would also proceed in a classical fashion, by decrypting the bundle, checking the signature and, if both operations are successful, applying the update. So far, everything works smoothly enough without any blockchain infrastructure as we seemingly have solved both integrity and confidentiality issues within our reference architecture (assuming of course that neither the manufacturer nor the devices - or at least the subset of them owning the cryptographic material - are compromised).

Having said that, a blockchain infrastructure (Figure 2) can help in terms of availability as a blockchain with a critical-enough mass would bring the following two properties:

1) *Persistence* in time of anything written to it e.g., once written into the supporting blockchain, a legitimate - signed - software upgrade towards a given device will remain there, unaltered ad vitam eternam.
2) The blockchain infrastructure may be assumed to be much more resistant to availability threats e.g., impersonations or Denial of Service (DoS), than the manufacturer's own infrastructure.

As an example, a device could periodically poll the blockchain by picking randomly one of its supporting nodes and checking whether or not one or more updates have been posted for it (in which case it applies either all of them or the last one, depending on the system design). In this context, a blockchain with access rights management would be a plus (in terms of system load optimization) but is not necessary in terms of security properties.

[1]Of course, the case where there is a single public/secret key pair shared among all objects is a subcase of the present one.

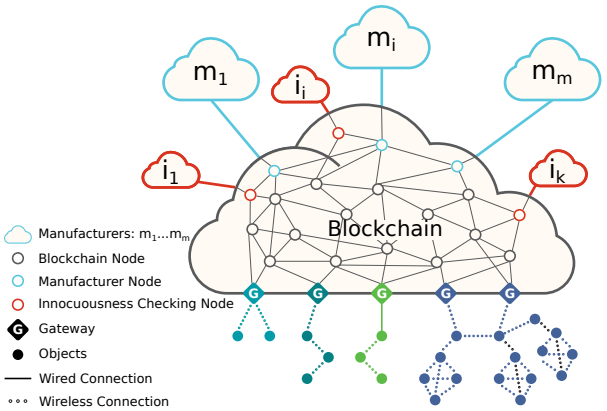

Fig. 2. Blockchain update architecture

Thanks to this architecture, it becomes possible to provide liveness guarantees on a software upgrade system.

Then, an attacker who would attempt a DoS on an IoT software update system e.g., in order to maintain an actionable vulnerability which would be fixed by an update, would be able to do so only via a DoS attack on the blockchain (which is presumed hard) or by flooding the blockchain with invalid software upgrades. Although the latter strategy may not be prevented by the blockchain infrastructure, it is not at all stealthy and can be detected by the manufacturer (whereas a man-in-the-middle on a manufacturer's own infrastructure would be both stealthy and unbounded in time).

It therefore appears that bringing a (massive-enough) blockchain into the picture would allow many manufacturers to share the burden of countering availability threats on their software upgrade infrastructures (to some extent it could be generalized to their overall communication infrastructure when availability is a stake). Per se, this is interesting as availability is all too often the Cinderella of the security pillars.

We define a manufacturer update transaction, corresponding to an update U, as the tuple $T = (T_{id}, U_{id}, U_v, U_{bin}, O_{type}, M_{id}, M_{sign})$ where T_{id} is a transaction identifier, U_{id} is an update identifier, U_v is an update version, U_{bin} is an update binary or an encrypted update binary, O_{type} is a device type, M_{id} is a manufacturer identifier and M_{sign} is M_{id} signature of all the previous fields. U_{id} can be simply the hash of U_{bin}.

Finally, note that when an update confidentiality is a concern, a manufacturer will have to encrypt an update binary U_{bin} of a transaction T before pushing it into the blockchain. Broadcast encryption mechanisms, such as those used for Pay TV, can be used to provide more efficient encryption of U_{bin} [5], [6].

D. Blockchain and Updates Innocuousness

In this section, we consider that manufacturers are not including update binaries into update transactions T i.e., $T = (T_{id}, U_{id}, U_v, O_{type}, M_{id}, M_{sign})$. When a transaction T

is pushed to the blockchain, it just serves to notify concerned devices that their manufacturer has a new software update available for download. However, devices will not download and install this new update until a set of innocuousness checking nodes have approved it.

Indeed, we extend the blockchain infrastructure by a set of k innocuousness checking nodes $I = \{i_1, \ldots, i_k\}$ (Figure 2). These nodes can belong to national cybersecurity agencies, or certifying and certified cybersecurity companies. These innocuousness checking nodes receive the new update binary U_{bin} directly from the manufacturer. Then, they not only check the integrity of the manufacturer signature of U_{bin} but also the innocuousness of U_{bin} itself against bugs and known attacks.

A device interested in a new update U will refrain from downloading U till at least $\lfloor \frac{k}{2} \rfloor + 1$ nodes of I have approved the innocuousness of U_{bin} and acknowledged it in the blockchain.

Let us assume that we have a complex system containing several devices coming from different manufacturers. One of these manufacturers may become malicious and install a malware in its device to spy on other devices or to collect end user private data. If the use of our blockchain architecture extended by innocuousness checking nodes is enforced by a global security policy, or even a law, such a malware will be detected by the innocuousness checking nodes. In addition, the latter will push a negative acknowledgment for this malicious update into the blockchain.

The advantage of using this approach compared to the one of the previous section III-C is mainly reducing the size of the blockchain by removing update binaries. However, extending the blockchain with k innocuousness checking nodes implies providing each device with k different public keys. These keys are compulsory for validating the acknowledgments of innocuousness checking nodes. However, keys management can become a disadvantage when k exceeds a certain threshold.

E. Update Acknowledgment

Once a device or an innocuousness checking node has installed an update and approved it, it must acknowledge its installation by sending a special transaction to the blockchain. We identify two types of acknowledgment transactions:

- We define a *positive* acknowledgment transaction as the tuple $A = (A_{id}, U_{id}, O_{id}, O_{sign})$ where A_{id} is an acknowledgment identifier, U_{id} an update identifier, O_{id} an acknowledging device or an innocuousness checking node identifier and O_{sign} is O_{id} signature of all the previous fields.
- We define a *negative* acknowledgment transaction as the tuple $(NA_{id}, U_{id}, O_{id}, MO_{id}, O_{sign})$ where NA_{id} is a negative acknowledgment identifier, U_{id} an update identifier, O_{id} an acknowledging device or an innocuousness checking node identifier, MO_{id} a malicious device or a malicious manufacturer identifier and O_{sign} is O_{id} signature of all the previous fields. Negative acknowledgment by innocuousness checking nodes of an advertised new update can serve as a metric for detecting misbehaving manufacturers (section III-D). Meanwhile negative acknowledgments referring to the same misbehaving device and coming from its peers can be used as a metric for insider attacker detection.

These acknowledgments are important as they permit to maintain a history of downloaded object updates, providing a simple accountability and logging system.

IV. Conclusion

In this paper, we investigated how a blockchain infrastructure can help in securing the deployment of updates for IoT devices. We enlightened the fact that, by design, a blockchain dramatically improves updates availability, due to the persistence and DoS risk mitigation properties. We refined our proposal to (i) allow extensive update innocuousness checking, by relying on trusted actors like national cybersecurity agencies; (ii) keep track of up-to-date devices; (iii) identify potentially malicious devices or manufacturers, therefore providing a building block for accountability.

For our future work, we consider several improvements. First, we will evaluate broadcast encryption mechanisms, to provide the confidentiality property at a lower cost, without having to deploy a dedicated update for each IoT device. Then, we will study if these mechanisms are compatible with resource-constrained devices, or if we have to tune these mechanisms to leverage lightweight cryptography. Second, we will investigate the use of threshold signatures in order to reduce the number of transactions transmitted by innocuousness checking nodes. Indeed, the latter will use a threshold signature to sign only one transaction to acknowledge the approval of an update instead of transmitting, at least, $\lfloor \frac{k}{2} \rfloor + 1$ acknowledgment transactions as currently proposed. Finally, it could be interesting to investigate which economic models can be built on this kind of distributed infrastructure; for instance, device manufacturers could give (financial) incentives to encourage device owners to retrieve updates by means of the peer-to-peer mechanism, in order to decrease the load on the blockchain infrastructure.

Acknowledgment

This work has been carried out in SystemX, and therefore granted with public funds within the scope of the French program *Investissements d'avenir*. This work is part of the projects *Environment for Cybersecurity Interoperability and Integration* (EIC) and *Cybersecurity of Intelligent Transportation Systems* (CTI). We would like to thank Jack Fazakerley, from SystemX, for helping with the figures.

References

[1] H. Tschofenig and J. Arkko, "Report from the Smart Object Workshop," RFC 6574 (Informational), Internet Engineering Task Force, Apr. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6574.txt

[2] H. Tschofenig, J. Arkko, D. Thaler, and D. McPherson, "Architectural Considerations in Smart Object Networking," RFC 7452 (Informational), Internet Engineering Task Force, Mar. 2015. [Online]. Available: http://www.ietf.org/rfc/rfc7452.txt

[3] D. Dolev and A. Yao, "On the security of public key protocols," in *IEEE Transactions on Information Theory*, vol. 29, no. 2, March 1983, pp. 198 – 208.

[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[5] A. Fiat and M. Naor, "Broadcast encryption," in *Annual International Cryptology Conference*. Springer, 1993, pp. 480–491.

[6] S. C. H. Huang and D.-Z. Du, "New constructions on broadcast encryption key pre-distribution schemes," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 1, March 2005, pp. 515–523 vol. 1.