Firmware Verification of Embedded Devices Based on a Blockchain

Boohyung Lee(✉), Sehrish Malik, Sarang Wi, and Jong-Hyouk Lee

Protocol Engineering Laboratory, Sangmyung University,
Cheonan, Republic of Korea
{boohyung,serry,sarang,jonghyouk}@pel.smuc.ac.kr
http://pel.smuc.ac.kr

Abstract. In this paper, a new firmware verification scheme is presented that utilizes blockchain technologies for securing network embedded devices. In the proposed scheme, an embedded device requests a firmware verification to nodes connected in a blockchain network and gets a response whether its firmware is up-to-date or not. If not latest, the embedded device can securely download and install the latest firmware from a firmware update server. Even in the case that the version of the firmware is up-to-date, its integrity is checked via the blockchain nodes. The proposed scheme guarantees that the embedded device's firmware is not tampered and latest. The effects of attacks targeting known vulnerabilities are thus minimized.

Keywords: Blockchain · Firmware verification · Embedded device

1 Introduction

According to the Gartner's report [1], the Internet of Things (IoT) era will change our live with network connected devices. The number of IoT devices is expected to be 5 billion by 2020 and the number will continuously increase. The IoT devices are tiny and small, while those are mostly embedded devices designed for specific operations, e.g., sensing, automation, etc.

Recent cyber attacks are targeting firmware, which is a software program on an embedded device [2], rather than services built on well turned servers [3]. Due to limited resources and capacities of embedded devices, strong security properties have not been applied yet to the embedded devices. Many bugs and vulnerabilities of embedded devices are reported every day and those are being used by attackers to break into the embedded devices.

One of feasible ways to protect the embedded devices is to reduce the attack window time by installing a latest firmware. It will help to minimize the effects of attacks targeting known vulnerabilities. As physical access to the embedded devices is possible, a verification of the firmware integrity is also required. In addition, due to the increasing number of the embedded devices, excessive network traffic may occur when downloading the latest firmware simultaneously

J.-H. Lee and S. Pack (Eds.): QShine 2016, LNICST 199, pp. 52–61, 2017.
DOI: 10.1007/978-3-319-60717-7_6

from a firmware update server. In other words, the current client and server model is not suitable for firmware distribution in an IoT environment.

With this in mind, in this paper, we propose a new firmware verification scheme that utilizes blockchain technologies. In the proposed scheme, an embedded device requests a firmware verification to blockchain nodes on a peer-to-peer decentralised network. It then receives a response whether its firmware is up-to-date or not. When the firmware is not latest, the embedded device can securely download and install the latest one from a firmware update server. Even in the case that the version of the firmware is up-to-date, the firmware integrity is checked via the blockchain nodes. Accordingly, the proposed scheme guarantees that the embedded device's firmware is not tampered and latest.

This paper is organized as follows. Section 2 reviews the ideas of blockchain. Section 3 presents the proposed scheme with the overall architecture and operation procedures. Section 4 concludes this paper.

2 Blockchain

The blockchain was first proposed in 2009 by an anonymous person, Nakamoto [4]. It was first used as a public ledger to provide trust transactions without an involvement of the third party for Bitcoin, which is a digital currency.

2.1 Block

In the blockchain, a block is used to preserve data or information. Every block contains a hash value of the previous block header that forms a type of chain [5]. It is then used to authenticate the data and guarantee the block's integrity.

The structure of block, for instance in Bitcoin, is made up of the block header and block body. The block header is composed of the block size, version, previous block header's hash, merkle root, etc. The block's body is consisted of the merkle tree and transaction. A merkle tree [6] is also called a hash tree. Leaf nodes of the tree make the hash value of blocks. It is useful because it allows an efficient verification of the block with the merkle root. Finally, a transaction is information of Bitcoin value that is broadcasted to the network and collected into blocks. In this scheme, the block is used with some changes in the block body.

2.2 Cryptographic Idea

A blockchain relies on two cryptographic methods: digital signature and cryptographic hash function. A digital signature is a way for demonstrating the authenticity of a digital message. It can be used to provide integrity and authentication of data as well as non-repudiation. A sender signs a message using the sender's private key. After a receiver receives this message, it verifies the message using the sender's public key. This message can be verified by anyone holding the valid public key of the sender [7].

A cryptographic hash function is a mathematical operation that computes a hash value. The function is deterministic, i.e., the same input will always produce the same output, with the following properties: pre-image resistance, second pre-image resistance, and collision resistance. In a blockchain, a SHA-256 hash function is used [8].

3 Proposed Firmware Validation Scheme

The proposed scheme provides secure operations to verify an embedded device's firmware. If the device's firmware is not up-to-date, the firmware update is proceeded with a firmware update server. Otherwise, the firmware's integrity is checked by blockchain nodes. Notations used are shown in Table 1.

3.1 Overview

Figure 1 depicts the overall architecture of the proposed scheme with the following entities:

- Blockchain node: A node in a blockchain network. A set of blockchain nodes is denoted as $B = \{b_1, b_2, ..., b_n\}$ and $b_i \in B$.
- Normal node: A normal node is a device which needs to verify its firmware in a blockchain network. A set of normal nodes is denoted as $N = \{n_1, n_2, ..., n_n\}$, $n_i \in N$, and $N \subset B$. It can be a request node or a response node. If a node requests its firmware verification, the node becomes a request node. After the verification process, it can validate its firmware. When a request node sends the request message to verify its firmware, other normal nodes can response. At this moment, the node responding to the request message becomes a response node that verifies the request node's firmware.

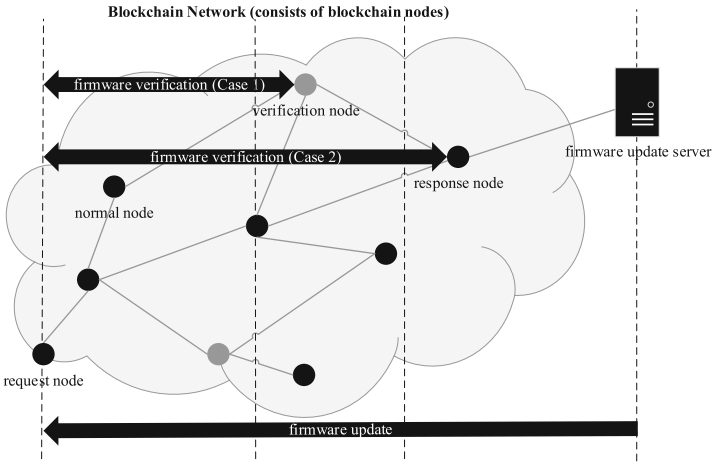

Fig. 1. Overall architecture

Table 1. Notation

Terminology	Definition
N	A set of normal nodes
V	A set of verification nodes
S	A set of firmware update servers
B	A set of blockchain nodes
D	A model name of n_i
r	Random number
ts	Timestamp
v	Current firmware version of n_i
v_{new}	Latest firmware version of n_i
v'	Current firmware version of n_j
fv	Current firmware file of v
fv'	Current firmware file of v'
fv_{new}	Latest firmware file of v_{new}
$H(fv)$	Verifier generated with fv
$H(fv')$	Verifier generated with fv'
$H(fv_{new})$	Verifier generated with fv_{new}
IDn_i	Identifier of n_i
IDv_i	Identifier of v_i
IDs_i	Identifier of s_i
E	Elliptic Curve
P	Base point of Elliptic Curve E
PUn_i	Public key of n_i
PRn_i	Private key of n_i
PUv_i	Public key of v_i
PRv_i	Private key of v_i
PUs_i	Public key of s_i
PRs_i	Private key of s_i
$SK_{n_i-s_i}$	Session key between n_i and s_i
$Sign_i$	Signature of n_i
$Sigv_i$	Signature of v_i
$Sigs_i$	Signature of s_i

– Verification node: A verification node is located to validate the firmware of normal nodes by a vendor. A set of verification nodes is denoted as $V = \{v_1, v_2, ..., v_n\}, v_i \in V$, and $V \subset B$. It has a verifier of latest firmware versions corresponding to the model name D. A size of the verifier is 256 bits when SHA-256 is used. The verifier can be updated periodically by the

firmware update server via the secure channel between the verification node and firmware update server.
– Firmware update server: A firmware update server may be administered by a vendor producing embedded devices. A set of firmware update servers is denoted as $S = \{s_1, s_2, ..., s_n\}, s_i \in S$. The normal node obtains the latest firmware from the firmware update server. The files (e.g., firmware) transferred between the two are encrypted via the session key.

Figure 2, wherein the request node is n_i and response node is n_j, shows the overall procedure of the proposed scheme. When a normal node wants to verify its firmware, it broadcasts a verification request message in the blockchain network. After receiving the message, any node (verification node or normal node) responds to the verification request. Following are the two cases depending on the type of the nodes involved.

– C1: Firmware verification between a normal node and a verification node
– C2: Firmware verification among normal nodes

Fig. 2. Overall procedure

In C1, a verification node confirms whether a request node uses the latest version of the firmware or not. If the request node uses the latest version of the firmware, it checks whether the request node's firmware is modified through exchanging and comparing each other's verifiers. Otherwise, the request node's firmware is updated by the firmware update server.

In C2, a response node asks other nodes to join the verification process. After receiving the messages from more than six other nodes, the request node and response node start comparing their firmware versions. If they use the same version of firmware, then they check whether the firmware has been altered by comparing each other's verifiers. If the version is not the same, the normal node requests an update from the firmware update server.

3.2 Assumptions

The followings are assumed for the proposed scheme.

- The network is composed of blockchain nodes connected to each other. More than eight normal nodes having the same firmware information with a request node should exist for the firmware verification without an involvement of a verification node. A vendor has to install more than one verification node.
- A normal node's ID is a random unique value as the ID is generated when the normal node requests the verification process based on its public key as like the address of the Bitcoin's blockchain. It provides privacy for the normal node.

3.3 Block Structure

The proposed scheme uses a different block structure compared with that of Bitcoin's blockchain. Note that we altered one field. The block in the proposed scheme is made up of the block header and verification field. The block header is composed of the block size, version, previous block header hash, and merkle root. The verification field consists of the verification counter, merkle tree, verification log, model name, firmware version, and verifier. Details of the field are given below.

- Verification counter: This is the number of successful verifications. It is a value which is only considered in a normal node's block. It is similar with the block height in Bitcoin [9]. In a verification node, this value is fixed at 0.
- Merkle tree: It is a tree information for the calculation of the merkle root. It is used for the verification of block data. It is also used as a feature for managing memory of node efficiently [4].
- Verification log: It is a verification log composed of the verification time (timestamp), request node's ID, and response node's ID. If a verification node responses for the request message, the request node's ID and verification node's ID are stored to this field. In addition, this field includes the signature of the request node using a private key so that all nodes can verify the signature.

- Model name: It is a normal node's model name.
- Firmware version: It is a normal node's current firmware version.
- Verifier: It is a hash value of a firmware file. This value is used to verify the firmware integrity without comparing genuine files. Each node stores a verifier in the verification field of the block. If the firmware file is composed of more than one file, then those should be concatenated before generating the verifier.

3.4 Cryptographic Ideas

The proposed scheme uses cryptographic schemes to encrypt a verifier to be transmitted, verify a signature, and exchange a session key.

- Data encryption and key exchange: When n_i requests firmware verification, n_i generates private key a, and selects random point of Elliptic curve P [10]. The public key aP is generated by multiplying a and P. The public key is used to encrypt a verifier. If n_i needs to update, its firmware is updated by the firmware update server. For this, Elliptic Curve Diffie-Hellman (ECDH) is used to generate the session key for encrypting the latest firmware file being transmitted from the firmware update server to the embedded device.
- Digital signature: When n_i signs its verification log using the private key of n_i, any node can verify this log with a corresponding public key.

3.5 Procedure

As mentioned, the proposed scheme has the two cases: C1 and C2. The case C1 is then divided into two sub cases (C1-1 and C1-2), while the case C2 is also similarly separated into C2-1, C2-2, and C2-3. The cases are set into motion after a node requests to verify its firmware. Hereafter we assume that n_i and n_j are the request node and response node, respectively.

C1. When n_i transmits the request message to the blockchain network and a verification node responds to the request, C1 starts. C1 has two different sub-cases.

- C1-1: It starts when n_i has the latest firmware.
- C1-2: It starts when n_i does not have the latest firmware.

When a verification node receives the request message, it checks whether n_i's firmware is latest or not. If n_i uses the latest firmware, then the firmware integrity is verified by exchanging each other's public key and comparing their verifiers. On the other hand, n_i requests the update operation to a firmware update server, which provides the latest firmware file, which is encrypted with a session key. Figures 3 and 4 show the procedures of C1-1 and C1-2.

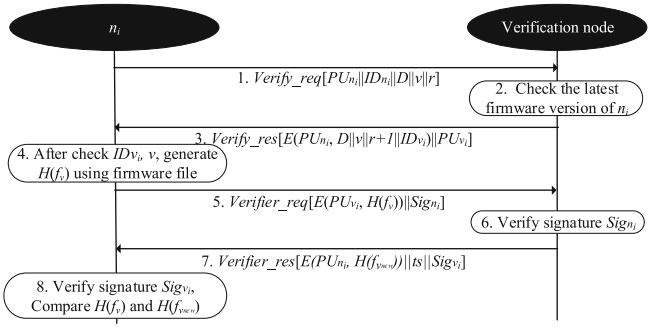

Fig. 3. Procedure of C1-1: n_i has the latest firmware

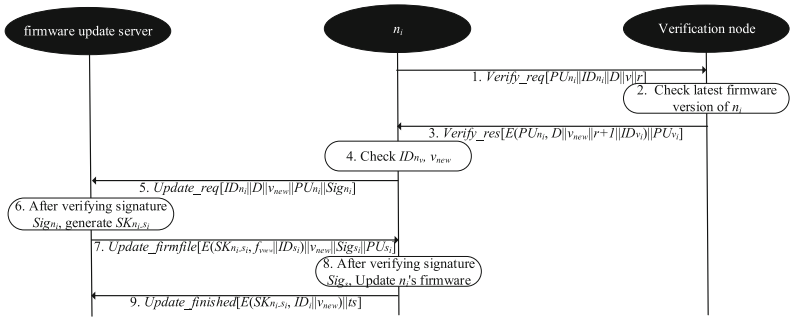

Fig. 4. Procedure of C1-2: n_i does not have the latest firmware

C2. The verification is performed among normal nodes in C2. Contrary to C1, n_j is not a verification node in C2. In C1, authenticity of a verifier is not considered since the verification node is managed safely by the firmware update server. However, in C2, the legitimacy of a verifier must be considered. To confirm the reliability of n_j's verifier, the Proof of Work (PoW) used in Bitcoin [4] is utilized in the proposed scheme. When n_j receives a request message, n_j asks other nodes to join the verification process by sending a join message. And other nodes perform the PoW stage. After completing the PoW, they add the request node's ID, response node's ID, and current time to the verification log in their blocks and then broadcast a verification log message to the blockchain network. If n_j receives the verification log messages from more than six other nodes, n_j responds with a request message of n_i. In this regard, the six nodes ensure n_j's verifier by collaborating with each other for this verification process. The C2 has three different sub-cases.

- C2-1: It starts when n_i's firmware version is higher than n_j's firmware version.
- C2-2: It starts when n_i's firmware version and n_j's firmware version are equal.
- C2-3: It starts when n_i's firmware version is lower than n_j's firmware version.

If each firmware version is equal, the verification process is performed as like C1-1. If one node has a lower version of the firmware than the other node, it requests updating its firmware to the firmware update server. Figures 5, 6 and 7 show the detail.

After the end of firmware verification, the request node makes its verification log including request node's ID, response node's ID, timestamp, etc. The request

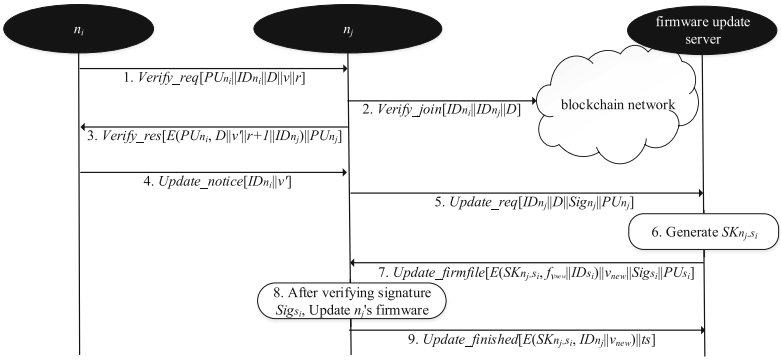

Fig. 5. Procedure of C2-1: n_i's firmware version is higher than n_j's one

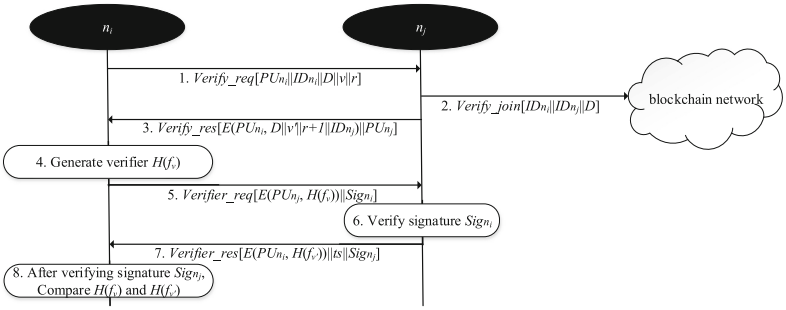

Fig. 6. Procedure of C2-2: n_i's firmware version is equal with n_j's one

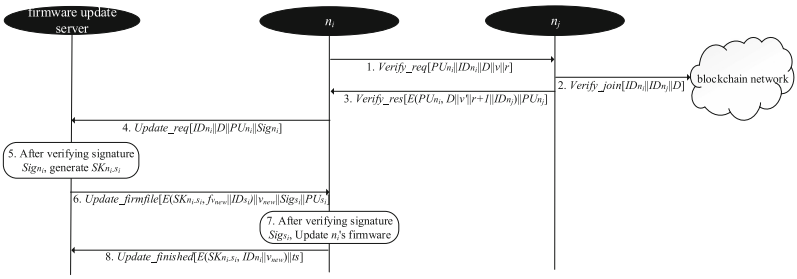

Fig. 7. Procedure of C2-3: n_i's firmware version is lower than n_j's one

node then broadcasts it to the blockchain network. Consequentially, it enables to validate the firmware integrity and decide whether the update firmware on the device requires or not without a user's control.

4 Conclusion

In this paper, we presented the proposed scheme that provides a secure firmware verification of an embedded device among blockchain nodes in the network. The proposed scheme guarantees that the embedded device's firmware is not tampered and latest. The effects of attacks targeting known vulnerabilities are thus minimized. For firmware updating, a firmware file is transmitted from a firmware update server to an embedded device. As a next work, we will study how to replace the client-server model for firmware file transmissions with a P2P network model.

Acknowledgment. This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (NRF-2014R1A1A1006770).

References

1. Gartner: Gartner Says 4.9 Billion Connected Things Will Be in Use in 2015. Gartner Newsroom, November 2014
2. Firmware - Wikipedia. https://en.wikipedia.org/wiki/Firmware
3. Choi, B.-C., Lee, S.-H., Na, J.-C., Lee, J.-H.: Secure firmware validation and update for consumer devices in home networking. IEEE Trans. Consum. Electron. **62**(1), 39–44 (2016)
4. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2009), Unpublished Manuscript
5. Blockchain Bitcoin Wiki. https://en.bitcoin.it/wiki/Blockchain
6. Hu, Y., Perrig, A., Johnson, D.B.: Efficient security mechanisms for routing protocols. In: Proceedings of the NDSS 2003, February 2003
7. Badev, A., Chen, M.: Bitcoin: Technical Background and Data Analysis. Federal Reserve Board, Washington, D.C. (2013)
8. Bider, D., Baushke, M.: SHA-2 data integrity for the secure shell (SSH) transport layer protocol. IETF RFC 6668, July 2012
9. Antonopoulos, A.M.: Mastering Bitcoin: Unlocking Digital Cryptocurrencies. OReilly Media, Sebastopol (2014)
10. Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., Moeller, B.: Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS). IETF RFC 4492, May 2006