






Article

Blockchain-Based Secure Device Management Framework for an Internet of Things Network in a Smart City

Seonghyeon Gong , Erzhen Tcydenova , Jeonghoon Jo , Younghun Lee 
and Jong Hyuk Park * 

Department of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul 08826, Korea

* Correspondence: jhpark1@seoultech.ac.kr; Tel.: +82-2-970-6702

Received: 27 May 2019; Accepted: 12 July 2019; Published: 17 July 2019



Abstract: The broadly configured smart city network requires a variety of security considerations for a heterogeneous device environment. Because a network of heterogeneous devices facilitates an attacker's intrusion through a specific device or node, a device management framework is required to manage each node comprehensively. This paper proposes a blockchain-based device management framework for efficient device management, scalable firmware update and resiliences on attacks against smart city network. This framework offers four device management and firmware update mechanisms based on the performance and requirements of each device: bidirectional mechanism of general end node and a unidirectional mechanism of the lightweight end node. This difference optimizes the resource of network and devices in terms of management and security. All management history of each device is stored in the blockchain and transmitting firmware between vendor and management node is conducted through a smart contract of blockchain for security and resilience on the attack. Through the framework proposed in this paper, the confidentiality and availability of device management on smart city network as well as integrity, auditability, adaptability and authentication for each node are ensured and the effectiveness of the proposed framework is presented through the security analysis.

Keywords: blockchain; smart city; device management; firmware update; internet of things

1. Introduction

The number of smart cities is growing every year as much the number of Internet of Things (IoT) devices connected to it and this trend will continue. The main goal of smart cities is the implementation of new technologies in all spheres of human life to make the functioning of urban life more efficient, thereby providing comfort and safety for citizens [1]. In a smart city, a huge number of different devices interact with each other and this gives more opportunities for cyber criminals to attack. Many types of devices make up a smart city, some of which are insufficient in protection and can easily be compromised. These vulnerable devices, along with management systems, are the primary attack targets. According to a Kaspersky Lab study [2], in the second half of 2016, about 40% of all industrial control systems were infected with malware. Moreover, there are already many cases of attacks on smart city management systems. In 2016, hackers successfully attacked Bowman Avenue Dam in New York. They took control of systems that could flood hundreds of homes in the area [3]. Transport networks are also vulnerable. In September 2016, it turned out that almost 25% of the systems of the San Francisco Municipal Transport Agency (SFMTA) are infected with an extortion program [4]. This malware opened the metro turnstiles and passengers were able to use the transport on weekends

for free. Also, in 2016, researchers showed how vulnerabilities could be used in traffic monitoring, data registration and processing systems: they hacked traffic control sensors in Moscow using a connection via a Bluetooth device, as a result of which they were able to control traffic lights [5]. In March 2018, a cyber attack on public computer systems in Atlanta paralyzed many of the city's functions, some for several months. The cost of recovery was almost 10 million dollars. Since smart cities directly deal with the lives of ordinary people, their safety should be above all. Any attack on it can entail not only monetary losses but also can serve as a threat to people's lives [6,7].

Therefore, you need to find a way to manage a smart city network composed of heterogeneous devices. Since an attacker can modulate the data generated in smart city terminal devices, the confidentiality and integrity of the data must be guaranteed and the integrity of the device itself must be ensured since the terminal device itself can be tampered with [8]. You should also consider the availability and latency of the services offered in the Smart city network while meeting security considerations. Finally, scalability and supervisory possibilities to accommodate the complexity of a smart city network must also be considered [9].

This paper proposes a blockchain-based device management framework for secure device management using a scalable protocol concerning the performance and resource limitation of each device nodes that constitutes a smart city network. The proposed framework monitors each the firmware status of each end node and checks whether the end node is hacked or tampered with by an attacker. The upgrade status of the firmware in each end node is analyzed and checked to keep it up-to-date. The monitoring information and firmware update histories are stored in the blockchain; thus, the device nodes and network can be effectively managed with scalability. The significant contributions of the study are summarized as follows. First, the proposed framework is designed for various security considerations such as integrity, confidentiality and availability of smart city network with blockchain. Second, this paper has illustrated the protocol flow charts to demonstrate the realization of blockchain realization in the smart city environment. Third, the security analysis of the proposed frameworks was conducted to measure the enhancement of various aspects of attack scenarios and security considerations.

The remainder of this paper is organized as follows. Section 2 introduces the related work on core technology, considerations and previous researches. Section 3 introduces the proposed blockchain-based framework architecture. In Section 4, the proposed framework is analyzed in terms of security and efficiency of the system to discuss the meaning and strength of the proposed framework. Section 5 conclude this research.

2. Related Work

2.1. Core Technology

2.1.1. Blockchain

Since blockchain is a distributed public ledger, it improves management by providing data verification such as digital content management and medical record tracking through blockchain. In general, a blockchain uses a different kind of blockchain depending on its use purpose and each has different characteristics [10,11]. Blockchain works with many elements such as network, transactions and authentication to store data. It is easy to integrate into various industries with these elements and the integrity of the blocks also ensures the integrity of the data [12,13]. Even if someone alters or forges the ledger based on one transaction information, it is virtually impossible to hack a large number of clients simultaneously because a majority of all users have that data [14]. Also, because distributed information is released to all members of the network through the distributed ledger and new information is updated in real-time, a blockchain has reliability and traceability of information. Relying on a distributed, peer-to-peer network approach that does not rely on existing centralized systems, it eliminates the intermediaries, increasing the efficiency and transparency of transactions and building a fast and secure network environment at low cost [15]. The blockchain enables network

members to validate transaction information and store it with the record quickly. In addition to transaction information, physical assets can also be converted into digital information and stored in a form that is easy to secure and supervise from forgery and alteration. Members can register new assets and change ownership in real-time without having to go through intermediaries such as banks, stock exchanges and payment providers. Also, existing complex and inefficient business processes can be simplified through smart contracts and basic rules for establishing contracts can be coded and recorded [16].

2.1.2. IoT for Smart City

A smart city is a network environment for solving traffic, environment and energy problems by integrating various information and communication technologies [17]. This network, which is based on communication technology, consists of objects, people and services [15]. The smart city uses various communication technologies such as 3G, 4G, LTE and Wi-Fi to connect various entities to make efficient use of resources and create various added value. Because of these reasons, the complexity of smart city networks consisting of complicated heterogeneous devices can cause a variety of problems [18]. A wide-ranging network increases the difficulty of management of end nodes and this can lead to an exponential increase in the point where an attacker can perform an attack [19]. Smart city technology aims to create sustainable cities that can solve various existing problems while creating added value.

2.2. Security Considerations of Device Management for Smart City

In this chapter, the security requirements for secure device management in a smart city environment is described.

Confidentiality Smart city technology collects, stores and analyzes data that could include sensitive information. Confidentiality makes sure that only authorized entities can reach that information [20]. Thus, confidentiality is one of the critical requirements that must be ensured in a smart city and the general technology to prevent unauthorized disclosure is cryptography. Since our proposed framework is based on blockchain, which is a technology that relies on cryptography using a private and public key, there is no suspicion about the confidentiality of the data.

Integrity To guarantee that data exchanged between entities is reliable and accurate, the integrity of that data must be ensured, which means that data is received at the destination without any leaks or breakage. Blockchain is designed to be resistant to modification of data through hash functions: once a data is added to the chain, it cannot be changed or deleted [21].

Availability Network management mechanisms for a smart city require high availability. Availability means that service or data is available when needed by authorized entities. A smart city cannot flourish without effective, real-time and reliable access to data. The proposed framework can provide availability due to the decentralized nature of blockchain; thus, there are no single points of failure.

Auditability Since every peer stores all transactions of the blockchain, any desired timestamped transaction or operation can be easily checked. Due to its transparency, the blockchain provides efficient auditability [22].

Authentication Authentication of all entities in a smart city must be ensured. Blockchain uses asymmetric encryption with a private and public key to ensure that the person accessing the data is the one whom he or she claims to be. Current IoT frameworks use centralized authentication that requires users to trust third-party entities for managing their data, thus using blockchain can provide secure authentication and deal with a single point of attack.

Latency The proposed framework uses private blockchain, which provides higher transaction bandwidth and lower latency in transaction processing.

Adaptability The heterogeneity of IoT devices significantly limits their interoperability. Blockchain-based network management framework will increase the degree of adaptability since blockchains are distributed databases independent of semantics [23]. Blockchain is capable of working

on heterogeneous hardware platforms, thereby allowing IoT frameworks to adapt to changing environments [22].

2.3. Previous Researches

Boohyung Lee and Jong-Hyook Lee [24] proposed blockchain-based secure firmware update for embedded devices in an internet of Things environment. In their proposed scheme each embedded device is a node of the network and it can send a request to other nodes to check if the device has the latest firmware. If it is not the latest, the device updates its firmware from other nodes via peer-to-peer sharing network.

Kristian Kostal et al. [25] proposed network monitoring and management architecture based on blockchain technology. Their proposed network has administrators that authenticated in the network by using digital signatures. Administrators modify the configuration of devices in the blockchain. After the configuration is added to the blockchain, all managed devices get informed. Then the device applies downloaded modifications by decrypting it using its private key.

Alexander Yohan et al. [26] proposed a PUSH-based update framework for IoT devices. Their proposed framework is designed based on the Ethereum blockchain platform. In this framework, the device vendor develops and stores a new firmware version in the firmware binary and then vendor node sends a newly created firmware update contract to the blockchain network. After that, the other vendor nodes of network verify a contract and the miner nodes collect and store it into the blockchain's ledger. The IoT gateways get this information, checks it and then downloads the binary file of firmware and forward it to the IoT device.

Boudguiga et al. [27] proposed application of blockchain where the device could periodically connect the blockchain by picking one of the nodes and check whether new updates have been posted from the vendor. And if so, the device downloads and installs it. When the vendor posts a new update, all nodes of the blockchain must verify it.

Samip Dhakal et al. [28] proposed a network for IoT device firmware update and integrity verification based on blockchain and the concept of delta update. In their proposed network, firmware files and information about it, including the version, checksum and metadata is stored in the private blockchain. To update a device the firmware manufacturer creates it and update service stores it in the blockchain. And then the firmware manager receives this information, queries the device for its information and forwards it to the blockchain for verification. The blockchain network checks the integrity of the firmware file and compares it with the firmware version of the device through their hashes. If it is not the latest, device firmware gets updated.

3. Blockchain-Based Secure Device Management Framework

3.1. Proposed Framework Architecture

The blockchain guarantees the integrity of transactions and behavior histories between nodes participating in the network with the level comparable to the security strength of the internal hash function such as SHA-3 [29]. Also, by executing a smart contract code which is implemented on a blockchain, nodes can execute reliable functionalities.

This paper proposes a new secure firmware management framework that has abundant resilience against malicious attacks on the security considerations of the smart city network. Beyond the framework proposed in previous studies, our research has been designed with scalability in mind, including lightweight devices with extremely limited resources. For these functionalities, the proposed framework handles the firmware update process of the devices constituting the smart city network as a smart contract. To this end, a blockchain platform capable of providing smart contracts such as Ethereum [30] can be used. The proposed framework consists of end nodes, management nodes, blockchain nodes and vendor nodes and vendor nodes store the firmware of their devices on the off-chain of the blockchain network or their server. The vendor nodes transmit the firmware to the

authorized management node by using the URL for downloading the firmware as the parameters of the smart contract. The blockchain network stores the firmware transmission history by the vendor node and the update history by the management node to the block. The most significant advantage of applying blockchain technology to heterogeneous network management is the ability to transparently manage the history of updates stored in blocks to quickly detect the attacker’s malicious act of injecting firmware. Smart city networks can be managed securely through periodic monitoring of end nodes and reliable firmware management through blockchain and smart contracts.

Figure 1 shows the proposed secure firmware update system architecture for device management and Table 1 explains notations in the proposed framework. The roles and functions of each node composing the proposed framework are as follows.

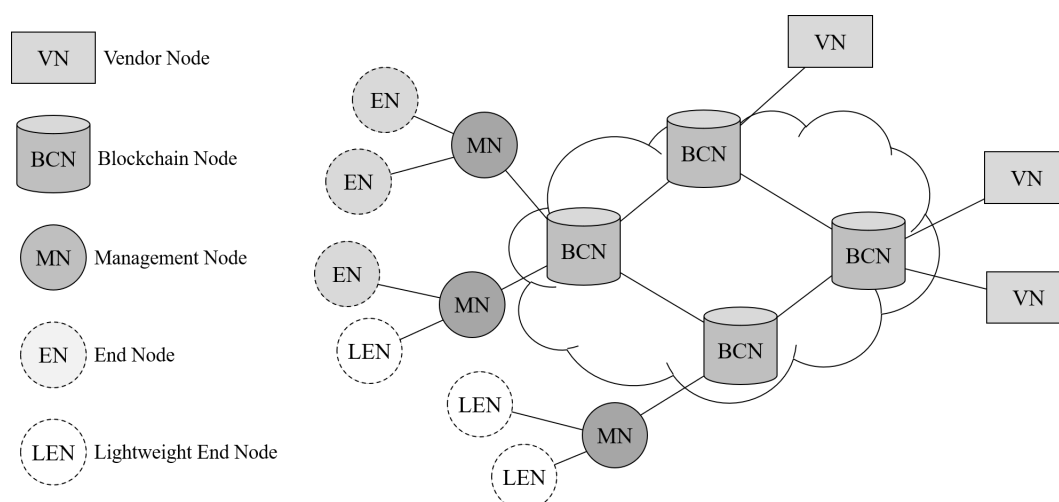


Figure 1. The proposed network architecture for secure device firmware update based on blockchain.

Table 1. Notations in secure device management framework.

| Notation | Explanation |
|-----------|---|
| ID_i | Unique identity information of node i |
| E_{S_i} | Symmetric encryption algorithm calculation which is used in node i |
| D_{S_i} | Symmetric decryption algorithm calculation which is used in node i |
| E_{A_i} | Asymmetric encryption algorithm calculation which is used in node i |
| D_{A_i} | Asymmetric decryption algorithm calculation which is used in node i |
| EK_{ij} | Symmetric encryption key which is shared between node i and j |
| DS_i | Digital signing using private key of node i |
| V_i | Digital signature verification using the public key of node i |
| PU_i | Public key of node i |
| PK_i | Private key of node i |
| H_i | Cryptographic hash function which is used in node i |
| VER_i | Firmware version of node i |
| fw_i | Firmware code of node i |
| S_i | Status information about firmware falsification of end node i |
| U_i | Update logs about end node i |
| FTI | Fixed monitoring time interval |
| RTI | Random monitoring time interval |
| UTI | Update time interval |
| r | Random number |
| $ $ | Byte data concatenation function |

- **Vendor node** The vendor node is a node that manufactured the end nodes. They set the smart city environment with end nodes and distribute updates and firewall rules for devices of a smart city network.

- **Blockchain node** The blockchain node is the entity of the blockchain network, which has the blocks that include the management information and logs for end devices. The management status, such as the hierarchical structure of and the update history of end devices, is stored in blocks.
- **Management node** The role of the management node is to intermediate between the higher entities and lower entities. Some devices which have enough computational power to fulfill the role of a small server such as IoT gateway could become management nodes. These nodes manage lower entities and supervise their status using symmetric and asymmetric encryption algorithms and hash functions. Since the management node manages a large number of heterogeneous end nodes, information about end nodes and environment information of each node should be stored. Therefore, the management node stores the unique identification information (ID_i) of the connected end node i , the version information (VER_i) of the firmware installed in i , the hash value of the firmware ($H(fs_i)$), the symmetric key and public key cryptographic algorithm and hash function specification to be used to communicate with i . The device management mechanism proposed in this paper considered that a separate lightweight cryptographic function might be used at the lightweight end node since the resource of some end node could be restricted dramatically.
- **End node** The end nodes, such as a camera, streetlight, various sensors and vehicles, are the primary functional devices of a smart city network. They could conduct basic functionalities for secure device management, including asymmetric encryption and have enough network traffic capacities. The end node includes asymmetric and symmetric encryption code and a cryptographic hash function code for communicating with the management node j in the installed firmware and stores a public key (PU_j) and unique identification information (ID_j) of the management node.
- **Lightweight end node** Some devices in the smart city network have extremely limited resources. These devices cannot conduct relatively heavy encryption algorithms like asymmetric encryption and can only conduct lightweight algorithms or optimized programs. General secure device management protocol does not fit this limited nodes and management procedures need to be managed by higher supervisor nodes. The lightweight end node i includes symmetric encryption algorithm and lightweight hash function code for communicating with the management node j in the installed firmware and stores a unique identification number (ID_j) and the hash value of connected management node's private key ($H_i(PK_j)$).

The proposed framework uses four different device management protocols to meet the security considerations of the heterogeneous network. These protocols are distinguished based on the computational performance of the device, that is, the ability to perform public key cryptographic operations.

3.2. Secure Device Management Protocol for End Node

An end node that has enough computation performance to sufficiently operate the secure device management protocol uses a bidirectional node management protocol to satisfy a high level of security strength. The bidirectional node management protocol consists of a node status monitoring protocol carried out by a management node, which is an upper node and a periodic update protocol, which is performed by an end node (lower node). The procedure of the node status monitoring protocol is as follows (Figure 2).

1. Status monitoring protocol occurs periodically. The management node waits for $t = FTI$ after the previous monitoring. FTI is a fixed time interval for the next monitoring and RTI is a random time interval to accept monitoring protocol request. These time intervals modulate the complexity of the network by adjusting the number of repetitive packets. Also, by staying in the idle state, end nodes can save the waste of resources and can ignore malicious communications such as

the distributed denial of service(DDoS) attack. FTI and RTI can be adjusted according to the performance of the device.

2. After waiting for $t = FTI$ from previous monitoring, the management node sends a monitoring process message to the device i under its management to check whether the firmware of the end node has tampered. The monitoring process message includes the ID of the management node and the temporal random variable r to prove that the sender is the legitimate management node. This message is encrypted by the private key of the management node and transmitted to the end node.
3. The end node receiving the monitoring process message decrypts the message with the public key of the management node assigned to it. Each end node has the ID information of connected management node, after decrypting the request, it can be verified through ID_j information comparison that is sent from the correct management node.
4. The end node which verifies the monitoring process message transmitted from the legitimate management node encrypts its firmware version, the currently installed firmware code, the firmware hash value and the temporary random variable r received from the management node to report its firmware status. The end node encrypts this message using the symmetric key cryptography algorithm with a pre-exchanged symmetric key.
5. The management node decrypts the message received from the end node using the pre-exchanged symmetric key. Using the r value, the management node can verify the legitimacy of the end node.
6. The management node verifies whether the firmware has tampered with the information sent from the end node by comparing with the version of the end node and the firmware hash value stored in its database. If the hash value sent from the end node is not valid, step 7 and 8 are omitted and abnormal status of end node's firmware is reported to the blockchain network through step 9.
7. When the integrity of firmware is confirmed, the management node sends new $FTI||RTI$ information to the end node. This information is encrypted with the pre-shared secret key using a symmetric encryption algorithm. This message updates the FTI and RTI status of the end node. By updating these time interval information, an external attacker cannot guess when they can send malicious monitoring request to the end node.
8. If the end node receives the FTI and RTI information from the management node, the end node updates its FTI and RTI values. This process is performed only when step 7 is performed.
9. If the monitoring process is successfully performed, the management node sends the ID and the status information S_i of the end node and hash value of firmware to a node on the blockchain network. At this time, the management node encrypts the data with its private key to prove the sender.
10. The blockchain node verifies the digital signature of the data using the public key of the known management node for the received data. If it is confirmed that the data is valid, the blockchain network stores the data in the block. Because the ID information of management and end nodes and the status of firmware are stored in the blockchain, all participants of this blockchain network can confirm the maintenance status of each node.

The device monitoring process initiated by the management node frequently checks whether the firmware of the end node has tampered. The frequency of confirmation depends on the computation performance and power consumption limit of the end node and the management node and the state of the end node firmware is stored in the blockchain network with the information of the management node. Therefore, all devices connected to the blockchain network can check the status of all nodes connected to the smart city network and provide information on which end node is being monitored by which management node. This information reduces the likelihood that an attacker injects malicious end nodes into the network.

An end node device that has enough capacity to operate the secure device management protocol can actively request a firmware update protocol to the management node. The request period of the update request protocol initiated by the end node could be varied depending on the size of the end node, the management node and the blockchain network. Figure 3 shows the detailed process of update request protocol.

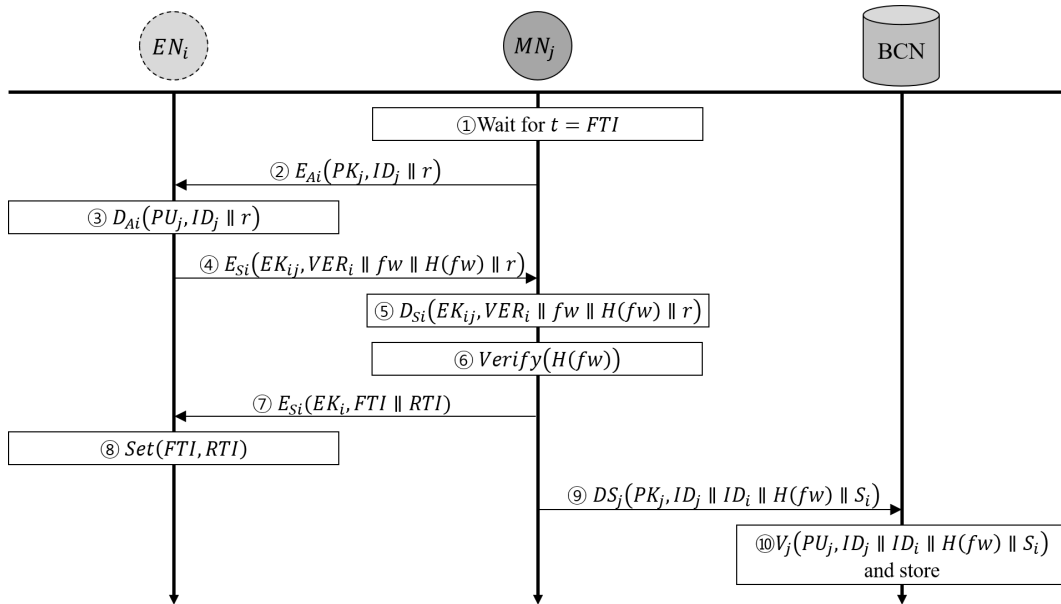


Figure 2. Protocol flow chart of monitoring process for end node.

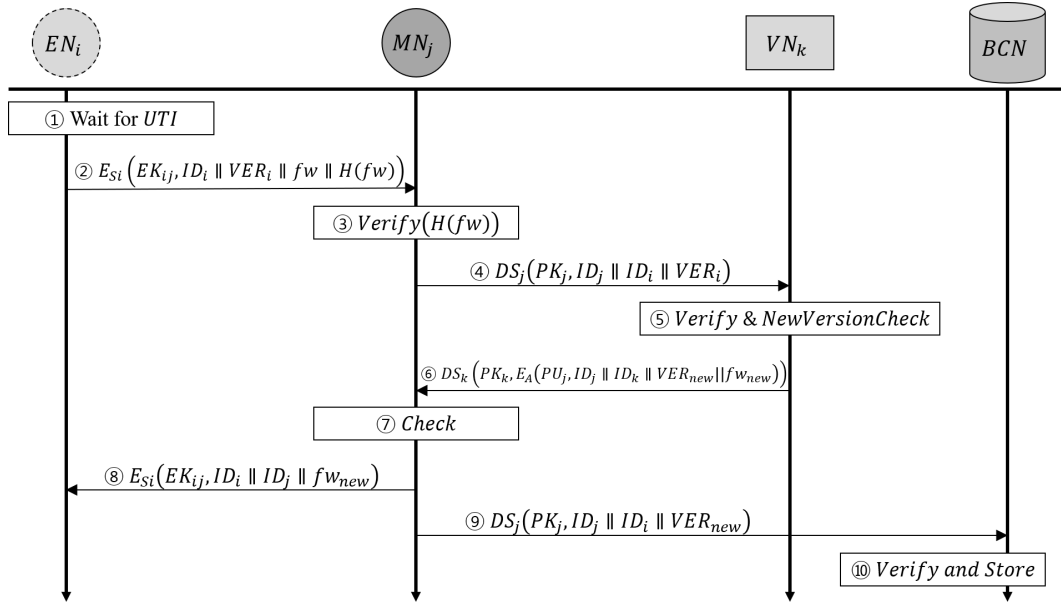


Figure 3. Protocol flow chart of update request process of end node.

1. The end node i initiates the process of periodic update request after it operates for UTI time. If the high level of security is required in an end node, the management node can adjust the update interval by reducing UTI value.
2. After the update request process starts, the end node sends a message to inform its firmware status ($ID_i || VER_i || fs || H(fs)$). This message is encrypted using a pre-shared symmetric key and sent to the management node.

3. The management node decrypts the received message and checks the firmware version currently installed on the end node by checking the hash value of the firmware code. If the hash value of firmware is not valid, the management node reports that abnormal status of end node to blockchain network through the message of step 10 in the monitoring protocol.
4. The management node provides the ID and version information of the end node to the vendor node and requests the latest version information of the end node product. This request message is signed with the private key of the management node.
5. The vendor node verifies the forwarded request from the management node and verifies the latest version of the end node product. If the firmware version of the end node is up-to-date, the vendor node sends only the version information and does not transmit the firmware code, then terminates this protocol.
6. The vendor node digitally signs the verified latest version information and the latest version of the firmware code with the private key of the vendor node. The encrypted message is encrypted again with the public key of the management node, so that only the requested management node j can decrypt this message.
7. The management node verifies the message delivered from the vendor node and compares it with the version information included in the request of the end node to confirm the existence of the update.
8. If there is a new update, the management node encrypts the new firmware file with a pre-shared symmetric key and sends it to the end node.
9. After performing step 8, the management node sends the update history of the end node to the blockchain network. This message is signed with the private key of the management node.
10. Blocks in the blockchain network verify the validity of the message with the public key and store the update history in the blockchain.

This regular update request protocol keeps the end node's firmware up-to-date. Using this protocol, smart city networks resist the attacks using outdated firmware.

3.3. Secure Device Management Protocol for Lightweight End Node

The presence of lightweight devices with extremely limited resources presents a huge security risk to the entire network. Devices in environments that cannot perform the basic algorithms required for secure communication and secure device management, such as public key cryptography systems, are exposed to a variety of attacks and threats. For lightweight devices, a dedicated security process is required and this process must not affect the operation of lightweight devices, such as the amount of power consumption. Figure 4 shows the device monitoring process for a lightweight end node.

1. Similar to the monitoring process of the general end node, the monitoring process for the lightweight end node also operates using a waiting time interval of a fixed time interval and a randomly specified time interval as long as $t = (FTI + RTI)$.
2. The management node concatenates the randomly generated random number r with the value hashed FTI information. This data is encrypted with the pre-shared symmetric key.
3. The lightweight end node decrypts the received message and performs a hash operation on the stored FTI value. Because the FTI value is the information that is synchronized by the management node, the lightweight end node can verify that the monitoring message is from the legitimate management node which has the same FTI value.
4. A lightweight end node that has verified the legitimate management node sends the device ID information, firmware code, the hash value of firmware code and $FTI||r + 1$ value to management node using encryption.
5. The management node can verify that the response was sent from the legitimate lightweight end node by checking the hash value for $FTI||r + 1$. The management node that confirmed the

- lightweight end node checks the value of $H(fs)$ to confirm that the firmware of the lightweight end node has not tampered.
6. The management node verifying the integrity of the lightweight end node sends an encrypted message to the lightweight end node to update the FTI and RTI values.
 7. The lightweight end node decrypts the received message and updates the monitoring process execution cycle using $FTI||RTI$.
 8. The management node sends a message to the blockchain network to record information about the integrity of the firmware of the lightweight end node. The message includes the ID of the end node and the management node, the hash value of the firmware and the firmware status information of the end node. This message is signed and transmitted by the private key of the management node.
 9. A node in a blockchain network verifies a message received from a management node with a public key of a management node to verify a message. The state information of the verified end node is stored in the blockchain.

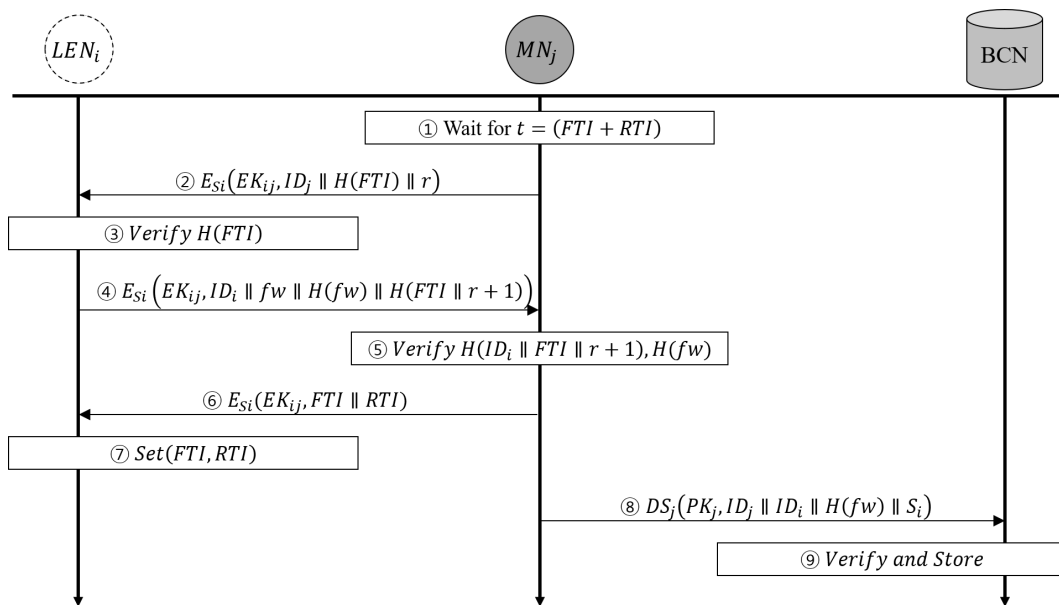


Figure 4. Protocol flow chart of monitoring process for lightweight end node.

3.4. Update Propagation Protocol

The vendor node periodically provides firmware updates. The new firmware produced by the vendor node is delivered to all the management nodes registered to the vendor node. Figure 5 shows the process in which new firmware updates are propagated to the end nodes by the vendor node and the update details are stored in the blockchain.

1. The vendor node signs the newly created firmware with its private key and delivers it to the management nodes.
2. The management node confirms the received new firmware with the public key of the vendor node.
3. Management nodes deliver new firmware to their registered end nodes. At this time, the management nodes transmit the firmware version information and the firmware hash values of the end nodes are stored in the database together, thereby proving the management node itself. The message sent by the management node is encrypted using a pre-shared cryptographic key with the end node.

4. The end nodes decrypt the message received from the management node, check the version information and the firmware hash value and confirm the management node. After the confirmation process for the management node is completed, the end nodes confirm the firmware update.
5. The management node that performed the update to the end nodes delivers the update history to the blockchain network. The message containing the update history is signed with the private key and transmitted.
6. The node in the blockchain network confirms the received message using the public key of the management node. The update history information that has passed the confirmation procedure is stored in the blockchain network.

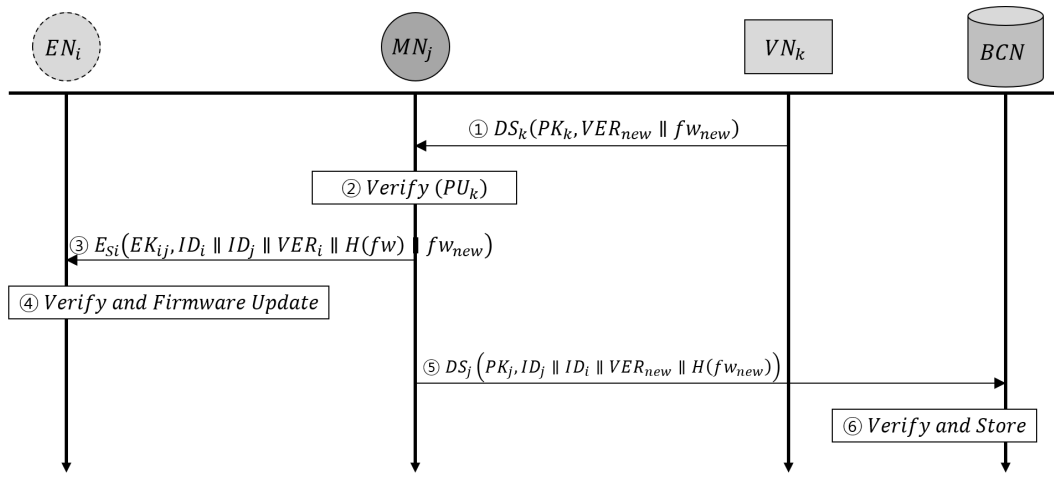


Figure 5. Protocol flow chart of update propagation process of vendor node.

3.5. Consensus Algorithm

The proposed framework uses proof-of-stake as a consensus algorithm in the process of executing a smart contract for firmware management and storing the monitoring results of management nodes to the blocks. The proof-of-stake is an algorithm designed to solve the energy consumption problem of the proof-of-work algorithm and to enhance the safety of the agreement algorithm. The proposed framework chooses the node to validate the block through the coin age-based selection mechanism. Each blockchain node uses its stakes to gain the right to validate the block. This mechanism prevents the entire blockchain network from being taken over by a node and a node that has proven an inaccurate block quickly loses its stake and right to validate the block. This proof-of-stake eliminates competitive mining processes, thereby avoiding unnecessary power consumption. The proposed framework provides a fair block validation opportunity for blockchain nodes and vendor nodes participating in a blockchain network; thus, one vendor node cannot take over the device management environment. In order to validate malicious blocks, an attacker must retain a stake of more than 51% of the entire network, which can be a challenging condition, depending on the amount of stake and its price. In order to form a blockchain network based on equity evidence, the stake must be mined in various ways in advance. The smart city network can allow participation in the network by selling shares to the vendor nodes that make up the network.

4. Security Analysis

4.1. Attack Scenario

IoT and Smart city network environments use different security specifications because resources of devices such as computing performance and battery size are different. In this network environment,

an attacker can set various attack scenarios according to the attack target device. Figure 6 shows the attack scenarios for the proposed Smart city network environment.

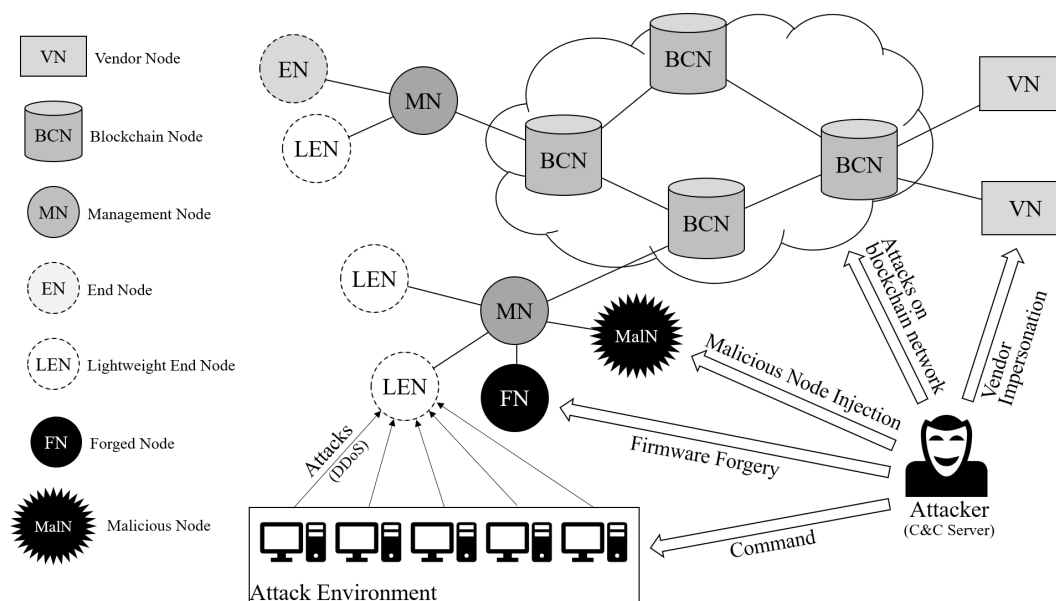


Figure 6. Attack scenarios on proposed smart city network environment.

DDoS attack A DDoS attack is a general attack technique that generates a large amount of traffic toward a specific node to compromise the availability of the device and the network. A heterogeneous network such as smart city uses various communication protocol specifications to communicate with each other. In the case of communication protocols for heterogeneous networks, such as CoAP [31], packet fragmentation and the presence of reflector nodes make it easy for an attacker to generate a large number of packets. In such a communication environment, an attacker can generate a large amount of traffic toward a specific node using a prepared attack environment.

Firmware Forgery An attacker can gain access to an end node and forge the firmware, thereby acquiring data or inducing abnormal operation. Forged firmware can compromise the confidentiality and integrity of data, the availability of services and an attacker can acquire illegal privileges on specific functions and requests through a forged end node.

Malicious Node Injection An attacker can gain access to services and networks by injecting the malicious end node he or she has configured into the network and making the node act as a legitimate node. A malicious end node recognized as a normal device from a management node can request data and obtain sensitive data provided by the service. Also, a malicious end node can send many requests to the management node to overload the management node and execute attacks that compromise the availability of the service and the network.

Attacks on blockchain network The management node generates all state information and management information for the end nodes and stores in the blockchain network. The management nodes report the normal and abnormal states of the end node's firmware to the blockchain network. The attacker can perform an attack that causes erroneous information to be stored during the consensus process of the blockchain network. If the number of nodes constituting the blockchain and the computation performance of each node is not sufficient, an attacker participating in the blockchain network may acquire more than half of the computation performance and guide the erroneous information to the block.

Vendor Impersonation The vendor distributes the new firmware through the management nodes and firmware distributed by a vendor node is propagated to all end nodes. If the attacker can impersonate the vendor node, he or she can distribute the abnormal firmware to every end node using the legitimate process of the protocol.

4.2. Discussion

This section describes the proposed framework in terms of security and efficiency. The security of the proposed framework is discussed considering the legitimacy of the device in the communication process and the efficiency of network expansion through resistance to DDoS attacks. Table 2 shows the differences of considerations on the framework between the previous works and the proposed.

Table 2. Comparison of considerations between the previous works and the proposed framework.

| Research | Consideration on Lightweight Node | Resilience on DDoS Attack |
|----------|--|--|
| [24] | X | X |
| [25] | Middleware is used to consider whether HTTPS support is available. | X |
| [26] | X | X |
| [27] | X | Resistance to DDoS that the blockchain has itself. |
| [28] | X | X |
| proposed | The proposed framework constituted the bidirectional and unidirectional management protocol for general and lightweight end node. Various cryptographic algorithms can be utilized in each end node based on its computation power and resource limitations. | By setting the time interval for the idle state of the end node, this proposed framework gives the resistance on external attacks. Also, frequently synchronized time interval makes it hard the attacker guess the appropriate time to initiate the attack. |

4.2.1. Security

Confidentiality A cryptographic algorithm with appropriate security strength should be used to ensure the confidentiality of communications. If each node can support encryption and digital signature process, that can be considered as a secure device. However, in a heterogeneous network environment, it is difficult for all devices to comply with such a system because of limited resources. The proposed framework proposes four different schemes according to the resource level of the device. A device with sufficient resources can perform both authentication and cryptographic communication between the end node and the management node. Lightweight end devices with insufficient resources can choose the lightweight cryptographic algorithm for device management process. A management node that manages lightweight end nodes can manage devices with various primitives by storing and managing the cipher specification of each device connected to it. All communications between the nodes in the proposed framework are encrypted and transmitted. Therefore, each communication process can provide minimum security strength through various cryptographic algorithms.

Integrity The integrity of the device as well as the integrity of the message is a critical consideration in the smart city and IoT environment. In a smart city environment that is physically broad and complicatedly connected with various devices, each device can be easily attacked by the attacker. Since access to all nodes cannot be controlled in practical points, each node must prove its suitability through the proof of its integrity. The proposed framework proves the integrity of end nodes through continuous monitoring of management nodes. The management node sends a periodic monitoring request message to the end nodes connected to it. All communications in device management protocol are encrypted using the cryptographic algorithm with pre-shared keys. Status of end node and managing logs of management node and vendor node are stored in blockchain with a digital signature. The Status includes the information about the firmware of the end node, hash value for integrity and version. And every relationship and management information among end node, management node and vendor node are recorded in the blockchain. This information ensures the integrity and suitability of each device and becomes a basis to detect the attack or malicious changes.

Authentication The authentication problem for the device nodes of the network can be solved through the public key cryptosystem, digital signature and the certificate. However, lightweight devices in heterogeneous networks such as smart city have a difficult to provide authentication capabilities adequately because of limited resources. The proposed framework includes a mutual authentication scheme between the end node and the management node on a lightweight device with minimal functionality. In monitoring the state of the lightweight end node, the management node encrypts and transmits the random number r generated by the random number generator. This random number is used with time interval information decided by the management node to authenticate each other using hashing. The attacker cannot get the original random number because he or she can only check the encrypted random number and the hashed random number on the network. In the authentication process between the lightweight end node and the management node, a hash value for the firmware code and the firmware code are used together. Since the lightweight end node does not use the update request protocol, even if the attacker fully grasps the lightweight end node, it is challenging to try malicious behavior. Also, since the management node synchronizes the *FTI* information, the effect on the entire network is small even if the lightweight end node is in control of the attacker.

4.2.2. Efficiency

Availability and Latency Availability and latency should be considered not only by the attacker's involvement but also within the system range of regular operation. An attacker can launch a DDoS attack on a node using prepared botnets or attack by packet amplification using the characteristics of the IoT communication protocol. If the packet length of the service data is long and a large number of packets are required to the end node or if there is a significant difference in the size of the service data between the nodes, the packet could be amplified. In a device management system that provides a firmware update function, an attacker can continuously send an update request to the firmware update server after infecting an end node or injecting a malicious node. Similarly, if the management node manages the end nodes, the attacker can exhaust the resources of the end nodes by impersonating the management node sending a persistent monitoring request to the end nodes. The proposed framework uses time interval(TI) information for resistance to these attack types. This time interval information is determined by the traffic processing performance of the management node and the resource of the end node. The role of time interval information is to prevent unnecessary traffic increase and the attack by rejecting the same request from the previous request for the specified time interval. The management node and the end node share the time intervals and can ignore the requests that occur within the time interval. To avoid this defense mechanism, the attacker can continuously monitor the communication cycle between the management node and the end node and then he can infer the time interval information using statistical techniques. However, the arbitrary period, *RTI*, gives some resistance to the attack through this monitoring and the management node periodically changes and synchronizes the time interval according to the given security strength. In a network environment requiring a high level of security, the time interval information changes very frequently so that an attacker cannot perform an attack using a statistical technique.

Auditability Auditing of end nodes in a smart city network is an essential function. The management nodes managing the end nodes periodically check the state of the end nodes and store them in the blockchain network. The blockchain networks not only store the state of all the end nodes but also include the management relationships between the end node and management node. The vendor node and management node managing the entire smart city network can confirm the resource allocation status and the abnormality of the network through the log stored in the blockchain. Also, since each management node stores security specification information for end nodes, it can quickly identify the weakest part in the entire network through the management nodes. This stored log allows for optimal management of the nodes' life cycle from a long-term perspective and prevents the

attacks on old nodes. The proposed framework provides a solution for device life cycle management of smart city by storing management information in the blockchain.

Adaptability The proposed framework provides various cryptographic specifications for adaptability to heterogeneous networks. AES [32] or RSA [33], the world standard encryption algorithm, provides a high level of security strength but it may be difficult to operate smoothly in environments with extremely limited resources. Lightweight cryptographic algorithms such as PRESENT [34], Simon and Speck [35] and lightweight hash functions such as SPONGENT [36] and LSH [37] could be suitable for this lightweight end device environment. In the proposed framework, the management node provides integrated management capability for various devices by separately managing the cipher specifications used in end nodes and lightweight end nodes. Also, some devices that lack security strength because of limited resources can be supplemented by adjusting the renewal cycle of the time interval.

5. Conclusions

As the internet of things developed, the difficulty of managing node device also increased. Especially for smart cities, a network of various devices in a wide area can cause many vulnerabilities due to the complexity of the network. However, the smart city network, which is directly connected to the national and social infrastructure, requires a high level of security due to its service characteristics. Therefore, a smart city consisting of heterogeneous devices must be secure, ensuring integrity, confidentiality and data availability. Also, authentication, latency, adaptability and auditability on node devices should be considered. The essential point of network management is device management. Each device should always be in a safe state and problems occurring on each device should be reported immediately. Centralized infrastructure-based frameworks may not achieve these goals and improper management and updating of devices can result in significant losses to smart city, requiring timely, secure and guaranteed device updates.

This paper has proposed a blockchain-based device management framework that can achieve the various security considerations on the smart city. The proposed framework uses private blockchain that consistently inspects the integrity of the device and stores the results device management. Given the heterogeneity of devices, the framework proposed four protocols for end node management. The first protocol is for end nodes with sufficient computation capacity to perform public key cryptographic operations and the second protocol is for lightweight end nodes with limited resources. The framework also proposed a bidirectional update protocol through an update request from the end device and an update propagation from the vendor node. The proposed protocols monitor the firmware of each device from time to time, share the information in a blockchain and provide a basis for instantly responding to security threats. This paper shows a framework for managing and operating a smart city network from a long-term perspective and can contribute significantly to the overall safety and availability of the entire network. This research can be extended to an environment that requires extremely high levels of service reliability, such as SCADA or national infrastructure in future research.

Author Contributions: S.G. and Y.L. conceived of the presented idea. S.G. and E.T. designed the proposed framework, performed the security analysis. S.G., E.T., J.J. and Y.L. discussed the related works, drafted the manuscript. J.H.P. supervised the research. All authors discussed the results and contributed to the final manuscript.

Acknowledgments: This study was supported by the Advanced Research Project funded by the SeoulTech (Seoul National University of Science and Technology).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Lytras, M.; Visvizi, A. Who uses smart city services and what to make of it: Toward interdisciplinary smart cities research. *Sustainability* **2018**, *10*, 1998. [[CrossRef](#)]

2. Garnaeva, M.; Sinitsyn, F.; Namestnikov, Y.; Makrushin, D.; Liskin, A. *Overall Statistics for 2016*; Blue Book; Kaspersky Lab: Moscow, Russia, 2016.
3. Jenab, K.; Moslehpour, S. Cyber security management: A review. *Bus. Manag. Dyn.* **2016**, *5*, 16.
4. Kang, W.M.; Moon, S.Y.; Park, J.H. An enhanced security framework for home appliances in smart home. *Hum. Centric Comput. Inf. Sci.* **2017**, *7*, 6. [[CrossRef](#)]
5. Saad, M.; Soomro, T.R. Cyber Security and Internet of Things. *Pak. J. Eng. Technol. Sci.* **2018**, *7*, 1–20. [[CrossRef](#)]
6. Elmaghraby, A.S.; Losavio, M.M. Cyber security challenges in Smart Cities: Safety, security and privacy. *J. Adv. Res.* **2014**, *5*, 491–497. [[CrossRef](#)] [[PubMed](#)]
7. Zhang, K.; Ni, J.; Yang, K.; Liang, X.; Ren, J.; Shen, X.S. Security and privacy in smart city applications: Challenges and solutions. *IEEE Commun. Mag.* **2017**, *55*, 122–129. [[CrossRef](#)]
8. Jeong, Y.S.; Park, J.H. IoT and Smart City Technology: Challenges, Opportunities, and Solutions. *J. Inf. Process. Syst.* **2019**, *15*, 233–238.
9. Suryani, V.; Sulistyono, S.; Widayawan, W. Internet of Things (IoT) Framework for Granting Trust among Objects. *J. Inf. Process. Syst.* **2017**, *13*, 1613–1627.
10. Sun, J.; Yan, J.; Zhang, K.Z. Blockchain-based sharing services: What blockchain technology can contribute to smart cities. *Financ. Innov.* **2016**, *2*, 26. [[CrossRef](#)]
11. Zheng, Z.; Xie, S.; Dai, H.; Chen, X.; Wang, H. An overview of blockchain technology: Architecture, consensus, and future trends. In Proceedings of the 2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, USA, 25–30 June 2017; pp. 557–564.
12. Krajjak, S.; Tuwanut, P. A survey on internet of things architecture, protocols, possible applications, security, privacy, real-world implementation and future trends. In Proceedings of the 2015 IEEE 16th International Conference on Communication Technology, Hangzhou, China, 18–20 October 2015; pp. 26–31.
13. Sharma, P.K.; Park, J.H. Blockchain based hybrid network architecture for the smart city. *Future Gener. Comput. Syst.* **2018**, *86*, 650–655. [[CrossRef](#)]
14. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: http://www.academia.edu/download/54517945/Bitcoin_paper_Original_2.pdf (accessed on 3 May 2019).
15. Sharma, P.K.; Kumar, N.; Park, J.H. Blockchain-based distributed framework for automotive industry in a smart city. *IEEE Trans. Ind. Inform.* **2018**, *15*, 4197–4205. [[CrossRef](#)]
16. Sharma, P.K.; Moon, S.Y.; Park, J.H. Block-VN: A Distributed Blockchain Based Vehicular Network Architecture in Smart City. *JIPS* **2017**, *13*, 184–195.
17. Kondepudi, S.; Ramanarayanan, V.; Jain, A.; Singh, G.; Nitin Agarwal, N.; Kumar, R.; Singh, R.; Bergmark, P.; Hashitani, T.; Gemma, P.; et al. Smart Sustainable Cities Analysis of Definitions; The ITU-T Focus Group for Smart Sustainable Cities. 2014. Available online: https://www.itu.int/en/ITU-T/focusgroups/ssc/Documents/Approved_Deliverables/TR-Definitions.docx (accessed on 12 May 2019).
18. Arasteh, H.; Hosseinezhad, V.; Loia, V.; Tommasetti, A.; Troisi, O.; Shafie-Khah, M.; Siano, P. Iot-based smart cities: A survey. In Proceedings of the 2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC), Florence, Italy, 7–10 June 2016; pp. 1–6.
19. Biswas, K.; Muthukumarasamy, V. Securing smart cities using blockchain technology. In Proceedings of the 2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Sydney, Australia, 12–14 December 2016; pp. 1392–1393.
20. Lin, H.; Bergmann, N. IoT privacy and security challenges for smart home environments. *Information* **2016**, *7*, 44. [[CrossRef](#)]
21. Dorri, A.; Kanhere, S.S.; Jurdak, R.; Gauravaram, P. Blockchain for IoT security and privacy: The case study of a smart home. In Proceedings of the 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kona, HI, USA, 13–17 March 2017; pp. 618–623.
22. Ali, M.S.; Vecchio, M.; Pincheira, M.; Dolui, K.; Antonelli, F.; Rehmani, M.H. Applications of blockchains in the internet of things: A comprehensive survey. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 1676–1717. [[CrossRef](#)]
23. Vanus, J.; Kucera, P.; Martinek, R.; Koziorek, J. Development and testing of a visualization application software, implemented with wireless control system in smart home care. *Hum. Centric Comput. Inf. Sci.* **2014**, *4*, 18. [[CrossRef](#)]

24. Lee, B.; Lee, J.H. Blockchain-based secure firmware update for embedded devices in an Internet of Things environment. *J. Supercomput.* **2017**, *73*, 1152–1167. [[CrossRef](#)]
25. Košťál, K.; Helebrandt, P.; Belluš, M.; Ries, M.; Kotuliak, I. Management and Monitoring of IoT Devices Using Blockchain. *Sensors* **2019**, *19*, 856. [[CrossRef](#)] [[PubMed](#)]
26. Yohan, A.; Lo, N.W.; Achawapong, S. Blockchain-based Firmware Update Framework for Internet-of-Things Environment. In Proceedings of the International Conference on Information and Knowledge Engineering (IKE), Las Vegas, NV, USA, 30 July–2 August 2018; pp. 151–155.
27. Boudguiga, A.; Bouzerna, N.; Granboulan, L.; Olivereau, A.; Quesnel, F.; Roger, A.; Sirdey, R. Towards better availability and accountability for iot updates by means of a blockchain. In Proceedings of the 2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Paris, France, 26–28 April 2017; pp. 50–58.
28. Dhakal, S.; Jaafar, F.; Zavorsky, P. Private Blockchain Network for IoT Device Firmware Integrity Verification and Update. In Proceedings of the 2019 IEEE 19th International Symposium on High Assurance Systems Engineering (HASE), Hangzhou, China, 3–5 January 2019; pp. 164–170.
29. Dworkin, M.J. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*; Technical Report; NIST: Gaithersburg, MD, USA, 2015.
30. Wood, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
31. Shelby, Z.; Hartke, K.; Bormann, C. *The Constrained Application Protocol (CoAP)*; Technical Report; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2014.
32. Rijmen, V.; Daemen, J. Advanced encryption standard. In *Proceedings of Federal Information Processing Standards Publications*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2001; pp. 19–22.
33. Rivest, R.L.; Shamir, A.; Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **1978**, *21*, 120–126. [[CrossRef](#)]
34. Bogdanov, A.; Knudsen, L.R.; Leander, G.; Paar, C.; Poschmann, A.; Robshaw, M.J.; Seurin, Y.; Vikkelsoe, C. PRESENT: An ultra-lightweight block cipher. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Vienna, Austria, 10–13 September 2007; pp. 450–466.
35. Beaulieu, R.; Treatman-Clark, S.; Shors, D.; Weeks, B.; Smith, J.; Wingers, L. The SIMON and SPECK lightweight block ciphers. In Proceedings of the 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 8–12 June 2015; pp. 1–6.
36. Bogdanov, A.; Knežević, M.; Leander, G.; Toz, D.; Varıcı, K.; Verbauwhede, I. SPONGENT: A lightweight hash function. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Nara, Japan, 28 September–1 October 2011; pp. 312–325.
37. Kim, D.C.; Hong, D.; Lee, J.K.; Kim, W.H.; Kwon, D. LSH: A new fast secure hash function family. In Proceedings of the International Conference on Information Security and Cryptology, Seoul, Korea, 3–5 December 2014; pp. 286–313.



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).