BLOCK-CHAIN BASED REMOTE UPDATE FOR EMBEDDED DEVICES

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Melike KAPTAN

December 2019
İZMİR

ACKNOWLEDGMENTS

I would like to thank my advisor Assoc. Prof. Dr. Tolga AYAV for welcoming me with this thesis work. His knowledge and vision guided me throughout this study.

My deepest gratitude is to my co-advisor, Prof. Dr. Yusuf Murat ERTEN for his continuous help and support. His encouragement was one of the prior inputs for me to finish this thesis.

Also i would like to thank all people who supported me. Especially the knowledge I got from Dr. Emrah Tomur was valuable and irreplaceable.

ABSTRACT

BLOCK-CHAIN BASED REMOTE UPDATE FOR EMBEDDED DEVICES

This research work is an attempt to devise a platform to send automatic remote updates for embedded devices. In this scenario there are Original Equipment Manufacturers (OEMs), Software suppliers, Block-Chain nodes, Gateways and embedded devices. OEMs and software suppliers are there to keep their software on IPFS (Inter Planetary File System) and send the meta-data and hashes of their software to the Block-Chain nodes in order to keep this information distributed and ready to be requested and used. There are also gateways which are also the members of the Block-Chain and IPFS network. Gateways are responsible for asking for a specific update for specific devices from IPFS database using the meta-data standing on the Block-Chain. And they will send those hashed secure updates to the devices. In order to provide a traceable data keeping platform gateway update operations are handled as a transactions in the second block-chain network which is the clock-chain of the gateways. In this study implementation of the two block chain shows us that, even though the calculation overhead of the member devices, with regulations specific to the applications block-chains provide applicable platforms.

ÖZET

GÖMÜLÜ CİHAZLAR İÇİN BLOCK-CHAIN TABANLI UZAKTAN GÜNCELLEME MODELİ

Teknolojide olan gelişmelerle birlikte birbiri ile haberleşen cihaz sayısı ve taşıdıkları fonksiyonalite sayısı artmaktadır. Bunların en göze çarpan örneği ise birbirleri ile ve dış dünya ile iletşim kuran araçlardır. Nitekim halihazırda kullanılan güncelleme mekanizmaları gelecekte birbiriyleriyle haberleşen cihazlar için, özellikle de üzerinde eskisinden daha çok yazılım barındıran araçlar için yetersiz kalacaktır. Bu tezin amacı, güncelleme platformlarına getirilecek olan olası yenilikleri göz önünde bulundurup gelişime açık bir model sunmak ve implementasyonunu gerçekleştirmektir. Bu sebeple, blok zinciri tabanlı uzaktan yazılım güncellemesi takibi yapan bir model düşünülmüş ve prototipi oluşturulmuştur.

TABLE OF CONTENTS

LIST OF FIGURES viii

LIST OF TABLES ix

LIST OF ABBREVIATIONS x

CHAPTER 1. INTRODUCTION 1
1.1. Motivation 2
1.2. Aim of Thesis and Contributions 3
1.3. Thesis' Outline 4

CHAPTER 2. BACKGROUND 5
2.1. Traditional Update 5
2.2. Block-Chain Technology 5
2.3. Block-chain Based Update in IoT 6
2.3.1. Block-Chain Based Update in Automotive 7

CHAPTER 3. RELATED WORK 9
3.1. Block-Chain Update Mechanism 9
3.2. Automotive Use-Case and Remote Update 11

CHAPTER 4. PROPOSED MODEL 13
4.1. Formal Model 13
4.1.1. Update File Storage 14
4.1.2. Transaction Handling 15
4.1.3. Mining Servers 15
4.1.4. Update 19
4.2. Scenario 20

CHAPTER 5. IMPLEMENTATION AND EVALUATION 25
5.1. Implementation 25
5.2. Evaluation 32

CHAPTER 6. CONCLUSION AND FUTURE WORK 38
6.1. Conclusion 38
6.2. Future Work 38

REFERENCES 40

LIST OF FIGURES

Figure **Page**

Figure 1.1. Block-Chain Based update basics for vehicles or other embedded units: Here the entities like OEMs are sharing a common platform which is a consortium block chain network which is a middle platform before the data is delivered to the devices via gateways 4
Figure 4.1. IPFS decentralized web 15
Figure 4.2. Transaction Visualisation For First Block Chain 16
Figure 4.3. Block Visualisation For First Block Chain 17
Figure 4.4. Merkle Tree Operation 18
Figure 4.5. Block Visualisation For Gateway's Block Chain 19
Figure 4.6. Sequence Diagram 24
Figure 5.1. Operation 33
Figure 5.2. Transaction Log 34
Figure 5.3. Transaction Log 34
Figure 5.4. Mining Log 34
Figure 5.5. Status Updates from Consortium 35
Figure 5.6. Gateway Mining Log 35
Figure 5.7. Gateway Mining Log2 35

LIST OF TABLES

Table **Page**

Table 5.1. Comparison of Features with Bitcoin Block Chain 36

LIST OF ABBREVIATIONS

OEM . . . . . Original Equipment Manufacturer
IoT . . . . . Internet of Things
POW . . . . . Proof-Of-Work
ECU . . . . . Electronic Control Unit
GW . . . . . GateWay
IPFS . . . . . Inter Planetary File System
p2p . . . . . peer-to-peer

CHAPTER 1

INTRODUCTION

Increasing popularity of Cryptocurrencies aroused curiosity for smart contracts and block-chain in recent times. Being able to generate trusted contract between accounts in an insecure environment makes great sense in academia. At last block-chains got more attention than the fields of crypto currencies and found applications also in (IOT) environment. Every other field like finance, social applications, automotive etc. intervenes people's lives and introduces a need to have a secure and authenticated data sharing mechanism.

Block chains are the chain of blocks and the block chain network is a network of devices that shares the same synchronized chain of blocks. Block chain network has protection mechanisms coming from its distributed nature. From the beginning to the present time those distributed servers are keeping the records of the data to be shared and operated. Hence, immutable records of the ledger are formed and protected under the verification mechanism of the members of the network.

Advances in the connected world requires secure communication between connected entities and robust servers serving those connected devices. With the increasing popularity of IOT, Internet of Things, devices in all areas of our lives started to communicate. This increasing communication introduces security and privacy threats. It is well understood that making things talk with other things is not feasible all the time, at least if not properly designed. Previously neither consumer electronics nor automotive products had hardly changing software. There was hardly a need for software update for a refrigerator or an embedded unit inside the vehicles. As the functionalities in sensors, actuators and similar devices advanced, companies paved the way for pervasive computing, and the software and hardware products started to gain separate understanding as a product in the development. It is usual now a days to buy a hardware which has a software on top of it getting updates often in order to enhance the functionalities or remedy the defects and bugs. The changing needs of the devices and systems force the environment to have built in precautions for security and privacy. The existing solutions to enhance the functionality relies mostly on centralized servers and introduces a single point failure. Therefore, the

current environment needs are well suited with the decentralized topologies.

The decentralized nature and the proof-of-work (POW) calculations which is an integral part of the block chains offer a solution to solve these problems for connected world and its applications, because, the increasing devices and their needs of connection requires more available servers in decentralized manner. The proof-of-work mechanism is another key element which makes the data corruption harder with the calculation overhead tradeoff. IoT might have the benefits of the block-chain networks with smart contracts or keeping fingerprints of their data in block-chains which will beat the data compromise. Another key benefit is having liable history of records when investigations are required for life threatening scenarios to be realized. Authorized data, disseminated and stored in network of computers promises for IoT and connected world.

In this thesis we propose a software update architecture for the connected devices. The proposed work covers most simple block-chain applications to distribute the updates.

1.1. Motivation

Connected world applications made devices more capable in terms of functionality, however it brought new challenges in security and privacy. Being able to share data unconventionally, not depending on a third party might be achieved by the Block chain servers. At least the data belongs to the devices, humans or organizations and there might be a world where they share their own entities without having to pay money to third parties or sacrifice security and privacy. Therefore, the conventional data distribution methodologies are open to creating a burden for companies and people and yet they are inadequate to ensure privacy, security and availability.

Cloud servers are good companions to keep and distribute the software to the devices on site like a vehicle after its sold or while it is in the production phase. But cloud mechanisms are the centrally managed entities and they do not ensure availability all the time. Also, there is a third-party supplier who can see the data and alter the data. In case of any liability inspections the cloud provider may also have a closer position to the one of the peripherals. Hence there is a requirement for suppliers and producers to keep the data in their own network without giving power of control to any third party.

Since the block chain networks offer a robust mechanism to keep the records secure and unchanged, its usages in the design of the data sharing platforms brings benefits

by itself.

1.2. Aim of Thesis and Contributions

The proposed study has contributions listed below:

- We propose to use a block-chain network to distribute software to the devices either to update them or to make installations at the production phase.
- In this architecture blocks cannot carry the update files themselves but their encrypted versions. So after putting the original files to the (IPFS)Interplanetary File System) the hashes of the files are used to create transactions.
- This block chain is used for keeping immutable records of all software producers in an authorized manner. Hence after releasing it to the consortium block chain network no producer can deny an improved functionality. Hence records are kept without changes, supporting integrity, with mining operation and synchronizing of the chain with the all its members.
- There are also gateway servers. Those servers have rights to read the consortium block chain but they cannot write to it. Whenever there is an update for a device that they communicate with, gateway servers will check the software updates in the block chain to decide if this device needs an update or not.
- Gateway servers will have another block chain of which they are members. This time the block chain will be keeping the records of the update operations. Whenever a gateway updates a device, it will send an update transaction. Hence another immutable record is kept to trace which device is updated by which server, which software it runs and who is the producer of this software.
- The block chain network of the gateway servers will be mining the Merkle tree produced from a bunch of transactions . Since their primary operation is sending the updates to the devices and as there are millions of devices there will be great amount of transactions ait is more efficient for them to be held in a Merkle tree and the root of the tree will be included in the blockchain.

1.3. Thesis' Outline

The rest of the thesis is organized as follows. In Chapter 2, conventional software update operations are given in 2.1 and Block chain technology described in 2.2. After that, block chain based update platforms are presented in 2.3. Since there is a gap in update of the futuristic vehicle solutions, block chain based update platforms for vehicles are investigated in 2.3.1. In chapter 3 we try to summarize previous studies in related with Block chain update in section section 3.1 and Block chain update in automotive use case in 3.2. In chapter 4 we give our proposed solution for update operation for devices. Lastly in chapter 5 we show implementation and comparison of our platform. Finally, we conclude the thesis with the advantages of designed model and discuss the future work in Chapter 6. In figure 1.1 the complete flow visualized.

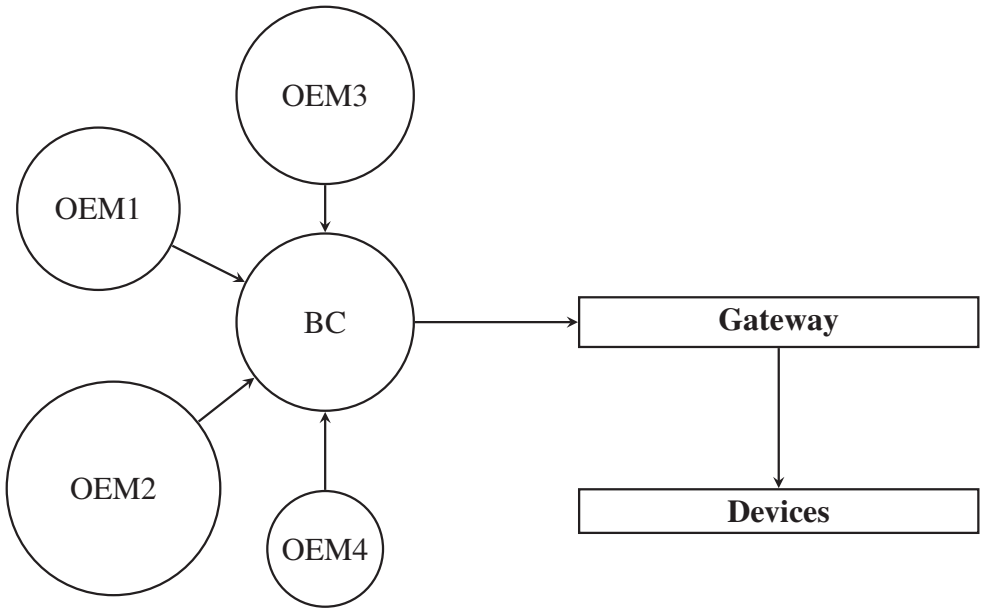

Figure 1.1. Block-Chain Based update basics for vehicles or other embedded units: Here the entities like OEMs are sharing a common platform which is a consortium block chain network which is a middle platform before the data is delivered to the devices via gateways

CHAPTER 2

BACKGROUND

In this chapter, the main ideas that the thesis is based on are explained separately.

2.1. Traditional Update

Updates for embedded devices are often done with physical connections on the site or with cloud-based methods. In these techniques software image files are flashed into the devices with an interface like a cable or air. In the former approach human technician must execute the work and the in the latter remote server performs this update via internet connection. The first method relies on human factor and the second introduces a single point of failure. Therefore, both methods have deficiencies. Those methods are quite acceptable in fields which do not require high security. For a smart home device like automatic shutter, if an update comes later than expected time because of some reasons like a technician to update was not able to come there on time or cloud server has issues and not serving for some time, people living in this smart home are not seriously affected. But in a field like automotive or road side unit production , if there is a bug in automotive software and needs to be corrected urgently, relying upon a service technician or one cloud service may have adverse effects on human lives.

2.2. Block-Chain Technology

Blockchain is the underlying structure behind the bitcoin (Nakamoto et al., 2008) and other cryptocurrencies. They have been taking remarkable place in these days from the time of first announcements done by Satoshi Nakamoto. For Satoshi Nakamoto it is a peer-to-peer electronic payment system which discards any third party from being a mediator and gaining remarkable amounts of money even from small transactions. He also denotes in the white paper (Nakamoto et al., 2008) that there must be a irreversible transaction system without so many information required from the customers. His peer-

to-peer trust-based transaction process treats every transaction as a signed entity by the sender adding the previous transaction hash and the next owner's public key. Thus, this chain of transactions for that coin creates a trusted process. The earliest transaction is also taken into account and all the transactions are broadcasted to the network, so all the nodes are aware of every coin.

Transactions are collected in a block and timestamped. The chained blocks are synchronized with the whole network. If one of the blocks or one block which includes a transaction which had been already placed in an already mined block, the network accepts the chain of blocks as the valid chain which is the longest. So, the mostly accepted order is the one to be accepted as the ultimate chain. In order to define the ultimate chain, one must have the chain which is accepted by at least 51 percent of the network, which is then considered as honest chain. And this shows that if an attacker can have 51 percent of the network he or she can deceive the network and this is only possible but known to be very hard to accomplish attack for blockchain.

The proof-of-work mechanism which is the essential part of the bitcoin, is applied to every block in order to reach a nderlinenumber of zeros in the hash value of a block. When this value is reached then the timestamp is given to that block in order to sort the blocks according to the chronological order. This number of zeros are defined according to the target value which is defined in order to compensate the increasing CPU speed of the hardware. This target is the average number of blocks required to be "mined" per hour. So, with calculated target value in a period time is determined by the initial number of zeros in the beginning of the hash value of the block.

Proof-of-work is one of the key entities in a blockchain which makes it impossible to change a record after its accepted by more than half of the network. The probability of making the fraudulent chain accepted is decreasing exponentially (Nakamoto et al., 2008) by time. Because an attacker after changing that one block accepted already, also must calculate the hash values of blocks coming after that one block since every block is chained to one another with the hash of the previous one. The probabilitiy of making fraudulent chain is calculated in (Nakamoto et al., 2008) with the random binomial walk and it proves that with the probabilty of the attacker will ever catch up from z blocks behind the last block at that time decreases exponentially.

2.3. Block-chain Based Update in IoT

Block-chains feature is being secure without requiring any centralized medium to build trust. What is missing in the IoT is the notion 'trusted environment'. So block-chain and IoT have a good match here. Data which is needed to be sent from one device to another must have its integrity, confidentiality and availability protected. Data needs not to be corrupted while in transit, needs to define its owner and needs to be secret in some fields and revealed in others. When we think about data shared between two IoT application based block-chain environment, it is definitely beneficial for preserving its integrity, since one cannot change data inside the blocks except under the one condition that the intruder has control over most of the nodes of the block-chain network. Even if he does, as stated before, he will have to modify all the blocks existing before the first corrupted block since each block caries the encrypted hash of the previous block. It is also beneficial in terms confidentiality where it has signed data by the owner of that data and beneficial for availability because data resides on a distributed environment and even if one node is down its not corrupted or lost. Every node in the block-chain network have an id, address, name and corresponding public and private key pairs. And the network has defined members, and each transaction broadcasted is signed by the private key of sender address.

Block-chain network keeps immutable records of the transactions. Those records are synchronized by the members of network. Each member is keeping the similar historical records of chain. This means block data is available even if some servers are down. Cloud environments are like hired third parties to keep your data. And trusting a third party may not be as trustable as using network power and utilizing all members of the network.

When we think of the liability, inspections are done by each environment involved in the process. Third party like platforms rely on a single party and are open to attacks by some intruders or biased inspectors. Increasing connectivity and increased attended parties makes the situation complex and trust less in such an environment. But data shared by different users shall be relying on only the unbiased network power with a block chain network.

2.3.1. Block-Chain Based Update in Automotive

Connected vehicle of near future is expected to be carrying a greater number of software applications than ever. These applications are installed onto a special kind of embedded devices called ECU: Electronic Control Unit. Approximately 30-40 ECUs are present in a middle range automobile today where this number will be increasing much more as vehicles connected to Internet will be carrying various applications in the future. When updates are delivered through the Internet, potential security problems will require researchers and manufacturers to apply proper protection mechanisms. One such precautions proposed in the literature against Denial of Service attacks is the use of Block Chain infrastructure where the code update sent by the manufacturers is distributed over various nodes of BC.

Vehicle firmware updates are a special subject as far as both the academia and the industry are concerned. Since automotive is a safety critical area, sending remote updates to the vehicles requires careful attention. The existing update methods include physical update done by a service technician. This the most common method. Since existing vehicles do not require to be updated very often in their lifetime, this methodology has served for years. With the advances in the connectivity and increasing importance of the connected vehicles, physical updates are losing importance and producers try to seek for new efficient methods. The number electronic control units found on a typical vehicle were much less than those in recent day vehicles because increasing functionality and connectivity requires more embedded units to process related functionalities. Today automatic parking, detection of road or road-side units, communicating with the vehicles riding side by side brings new concerns on the vehicle units update process. This means over-the-air updates become prominent.

Autonomous vehicles require updates more often for each functionality improvement or new developments, the industry might start to behave in a similar way to how our mobile applications are delivered. Wi-Fi or mobile networks will be beneficial in here.

Another point which over-the-air updates will bring is the liability solutions which has importance as well. Increasing connections must be traceable in case of any unwanted situations. In this kind of world there will be many suppliers and many more systems like computer vision, decision making etc. Whenever that software is distributed through the roads, bugs must be traceable especially whenever an incident happens like the case of fatal self-driving car accident (David Shepardson, 2019).

CHAPTER 3

RELATED WORK

In the literature, there are several works proposing to use Block-Chain for IoT and automotive-related tasks. These tasks include payment mechanisms for car rental, vehicle-to-vehicle and vehicle-to-everything communication, car sharing, insurance of vehicles and supply chain management. Since the scope of our study is on software update mechanisms, we include only such works in our literature survey section.

3.1. Block-Chain Update Mechanism

The demand for security in IoT and the connected world is increasing and it prevented the designed studies to be applied in real life, and security mechanism designs for connected world has gained attention. Block-chain attracted some designers and the number of block-chain based data exchange platforms designed has increased.

The starting point of our proposed design is the idea that there may be a universal way of updating connected devices in order to beat the shortcomings occurring because of lack of confidentiality, integrity and availability properties as clearly stated in (Boudguiga et al., 2017). This design is one of the first examples that are proposing to use block chain to send updates to resource constrained devices. In this work the authors also designed an additional mechanism for innocuousness check, assigning some of the nodes to some agencies which are responsible for checking integrity and removing bugs etc. This model is an example design of over-the-blockchain update model in order to beat problems in the way of the confidentiality, availability and integrity. However, this study proposes to put fingerprints of the update image files into the blocks, notifying devices about new updates but not stating how those devices will get the updates. In our study we are planning to create another instance of transaction model which will keep details about which devices are updated, when, by which server and version of the software created by which producer.

In (Dorri et al., 2016) authors presented a secure platform for smart home use-case. In their study, authors' design has a local block-chain, an immutable ledger in the smart home which is responsible for collecting all the communication transactions between sen-

sors, actuators or other smart devices. Rather than having a proof-of-work concept in their design the local block-chain keeps all the transactions handled by the devices. Also since the devices are resource constrained they need to store the records of the local blockchain in a cloud server which will create additional need for availability. In this study local blockchain makes the communication faster and the permission to communicate with any device is also managed by the overlay block-chain network. In the outer environment of the smart home, in overlay network, every smart home is a member of one overlay network at a time. The overlay network cluster heads share data with any other block-chain but since every cluster-head decides which data to keep, block-chain may have many forks. Also, local storage miner in each home sends its data to a cloud network in case of any data to send to the software provider for any feature extraction, which is a centralized entity. In their following works(Dorri et al., 2017) authors measured the time and packet overhead and energy consumption of the design they organized (Dorri et al., 2016). In their study they analyze the design and deduce that network and energy consumption of the block chain based smart home application required to be improved. In order to have an enhanced application the same authors designed the study (Dorri et al., 2017) and they changed the way they applied overlay network outside of the smart homes, while they keep the same private block-chain in smart home miners. Rather than having a peer-to-peer and forked block-chain in the overlay network they designed a public block-chain adoption and a trust mechanism. This trust mechanism is structured using a verification of the blocks according to the trust metrics of the miner instead of using Proof of Work mechanism in traditional block-chain. The authors in this study draw attention to the resource requirement, scalability and high delay challenges regarding block-chains and try to overcome them. These mentioned studies are good examples of access control but our aim is to design traceable flow of data.

In (Lee and Lee, 2017) another design for usage of block-chain in smart homes is presented. In this work the transactions serve for the transfer of the firmware update processes of the IoT devices. In this design the software provider company propagates the repositories of the newly arrived firmware updates among all the nodes of the block chain. There are gateways which are responsible for sending the updates for the devices it manages. Before the update operation there is a two-way requirements validation between the passive node, which is informing the gateway about the new update and the requirements, and the gateway. We have very similar handling for the software update sending mechanism to the devices, however, we have an additonal mechanism to provide

liable information from the gateway servers.

3.2. Automotive Use-Case and Remote Update

Blockchain based update studies in automotive requires more effort since it is a safety critical area. As there are predefined traditional ways of updating software on a vehicle, there must be safely applicable methods of in order to compete with these steady methods. In (Steger et al., 2018) authors use a wireless update methodology (Steger et al., 2016) and distribute the wireless software updates using block-chain. Their block chain-based update distribution model is the same model that they described in (Dorri et al., 2017). Software created by the suppliers and manufacturers is propagated through an overlay network in order to be sent to the vehicles. This work is both a futuristic and efficient example of block chain based updates in automotive industry. In their study authors compare the results of their mechanism to distribute the new updates and installing them to the vehicles with the wireless updates given in (Steger et al., 2016). Their results show that software distribution process requires remarkably less time than the local installation to the vehicles. And their comparisons regarding the certification based wireless update methods shows similar trends with respect to the packet and time overhead with increasing number of vehicular interfaces (Steger et al., 2018).

The same authors in (Steger et al., 2018) designed another block chain-based model in (Oham et al., 2018). In their study authors draw attention to the liability attribution for an environment with autonomous cars and a connected world. Their aim is to find a generic resolution for the issues related with Autonomous vehicles. When the autonomous vehicles are the the point in question from pedestrians to the big vehicles on the roads, they have potential to be included in any kind of dispute. Like in the case (David Shepardson, 2019) there is a need to find a liability model for such cases. Authors of (Oham et al., 2018) emphasize that increased connectivity requires more complex and untampered liability attribution model. In order to realize this untampered liability model Block-Chain based recording mechanism is proposed and improved to provide a decision giving mechanism for insurance companies.

Another study proposes to use Block-Chain network to send and get updates is (Baza et al., 2018). In their study authors think Autonomous Vehicles get updates directly from the producer and they also act as gateway nodes which distributes updates for

other AVs. Two mechanisms provide secure distribution of the updates, zero knowledge proof and attribute-based encryption. In this study smart contract are leveraged as the key elements to increase the repudiation of the distributor AV. In the example they used (Greenberg, 2015) to support their study, the company (Company X) recalls 1.4 million vehicles back because of the bug in the ECU software. This shows a big gap in terms of security when it comes to the vehicles. Software on the cars is enhancing and beside the requirements to keep that software secure and software updating mechanisms are also under risk.

To sum up we surveyed various kinds of software update mechanisms which are proposing to use block chain networks in the literature. We try to find a way to realise such updates in a method which is not complex and covers all mentioned concerns. Our aim here is to propose a model which provides integrity, confidentiality, availability at the same time supports liability of the data. Comparing with the literature survey our model provides a whole picture with a traceable approach in every key entity of the model.

CHAPTER 4

PROPOSED MODEL

In this study we try to introduce a secure and anonymous environment in order to distribute software through devices. As described before, there will be one block-chain which is used by the software providers of the embedded electronic units and another which is used by the remote software distributing units, namely Gateways, for those embedded electronic devices. Both chains will be storing transactions. Security comes from the distributed environment, which is robust to the attacks traditional servers are exposed to. Anonymity comes from the information in those chains which is the hash value of the image files. The roles and the detailed principles will be presented in the proceeding sections but before doing so, we will summarize the overall mechanism. below Any software provider or producer company, who wish sending its update or first installation to the devices, loads its produced image file to any platform which returns a hash value which can be used to search the image. With this returned hash the developer in the company creates a signed transaction with company?s private key including this hash value and the additional metadata field and sends it to the blockchain. Additional field indicates target device types that software will be installed. This transaction broadcasted to all miners of the software provider blockchain. Gateways will be informed whenever a new update transaction is broadcast to the producer?s blockchain. Then gateway starts to observe the mined version of the transaction.

This means that gateways have rights to read the blockchain data but cannot write. Having read the transactions inside the blocks of the chain, a gateway now has responsibility to distribute the image file to corresponding devices. There will be another block chain among gateways. Blockchain of gateways is storing all the information regarding the management of the update transactions. So, there will be signatures of sender gateways and exact timing of the update send operation. The mathematical expressions regarding this process will be given in the next section.

4.1. Formal Model

Original Equipment Producer (OEMs): Original Equipment producers are the producer companies of the embedded electronic units.
Software Providers(SP): Software Providers are the stakeholder companies providing software for OEMs.
Gateways (GW): Gateways are the server devices responsible for checking announced updates for devices and sending them to the devices
Devices (D): Embedded devices waiting to be updated.

$DeviceNo_i$=<(ProducerNo, Model, Number)>

Transaction (Tr): Transactions are the information that are sent to the miners via broadcast messages in order to be put inside blocks. In this way the sender and authorized information is shared.
Blocks (B): Blocks are the like the storage containers to keep possible number of transaction with the suitable nonce and timestamp.
Mining: The valid block comprises of a suitable nonce, a timestamp, bunch of transactions, authorized information and id. This suitable nonce is formed by a hash calculation using brute force trials. These trials are performed for ensuring the validity of the block.

4.1.1. Update File Storage

Before creating a transaction, software image files are kept in a file system. Every file has an encrypted version pointing to the address of those files which will be distributed to the blockchain network to be stacked in a mined block. IPFSis suitable for this purpose because its secure, easy to share files in an censorship-resistant web like structure. IPFS is a peer-to-peer distributed file sharing system (Benet, 2014) which is like a content based web instead of traditional location based web and its p2p nature makes it fast and secure. It uses an overlay network which has members which do not need to trust each other. IPFS has version control mechanism supported by the Merkle trees. These functionalities make IPFS a nice versioned-file-system like Git. Like in figure 4.1, decentralized servers of the IPFS network can ask contents from the nodes which are closest to it.

Block-chain mining operation is a heavy process because of its computation load.

This makes block chains unsuitable for storing files. The content of the software image file will be broadcasted through the block-chain members in order to be in a possible block. And the actual version of the files will be stored on the IPFS.

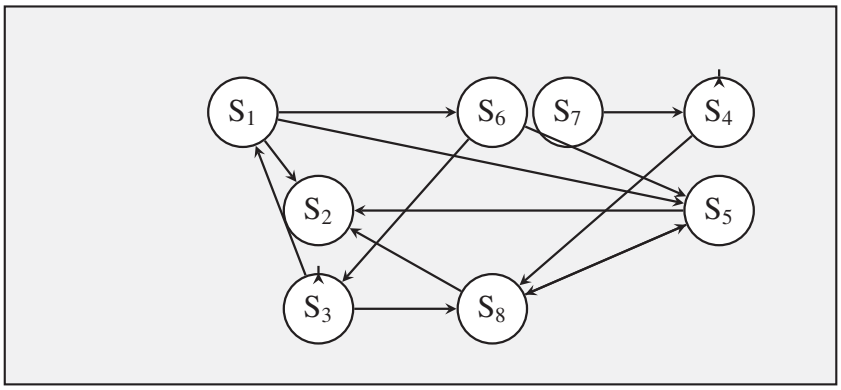

Figure 4.1. IPFS decentralized web

4.1.2. Transaction Handling

In the producer block-chain network each transaction sender has one public one private key. Transaction data is sent by the OEMs or Software providers indirectly. Whenever transaction forming data is completed and decided to be released through the provider Block-Chain, other information is added to the transaction such as operation time and signature of the sender and public key of the sender. Then,

$Transactioninformation =< (SenderID, Imghash, modelNO) >$

Turns into a transaction,

$Tr =< SenderID, Imghash, modelNo, timestamp, PublicK, signature, TransactionId >$

By the time the transaction is produced, its ready to be broadcasted to the entire producers Block-Chain network for mining operation.

Whereas for the Gateways Block-Chain all the fields will be used for to the identification of the Gateways. The sender will be the Gateway and the public key will be the gateways public key and the hash of the data will refer to the update image of the sent update etc.

4.1.3. Mining Servers

Following the broadcasting of the transactions the members of the producer Blockchain put the transaction into a queue. The mining mechanism will be collecting the transactions in this queue and place them into a block. This time Block identification fields will be identified. Each block will be keeping hash of its previous block, its id and number of transactions selected to be placed into that block as shown figure 4.2. Blocks will be chained together from the very first block which is genesis block, a block which has empty transaction field and has block id = 0. The blocks mined later will be aligned after the genesis block if they are accepted as a valid block. A sample block chain for the producers chain of blocks can be seen in figure 4.3.

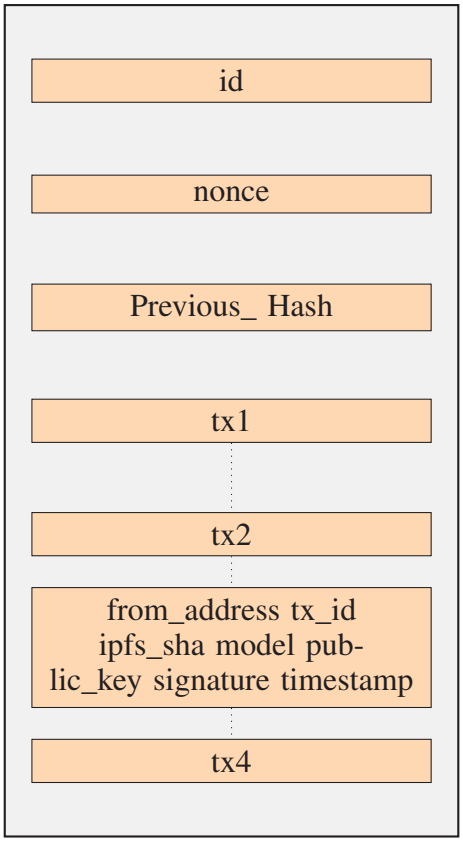

Figure 4.2. Transaction Visualisation For First Block Chain

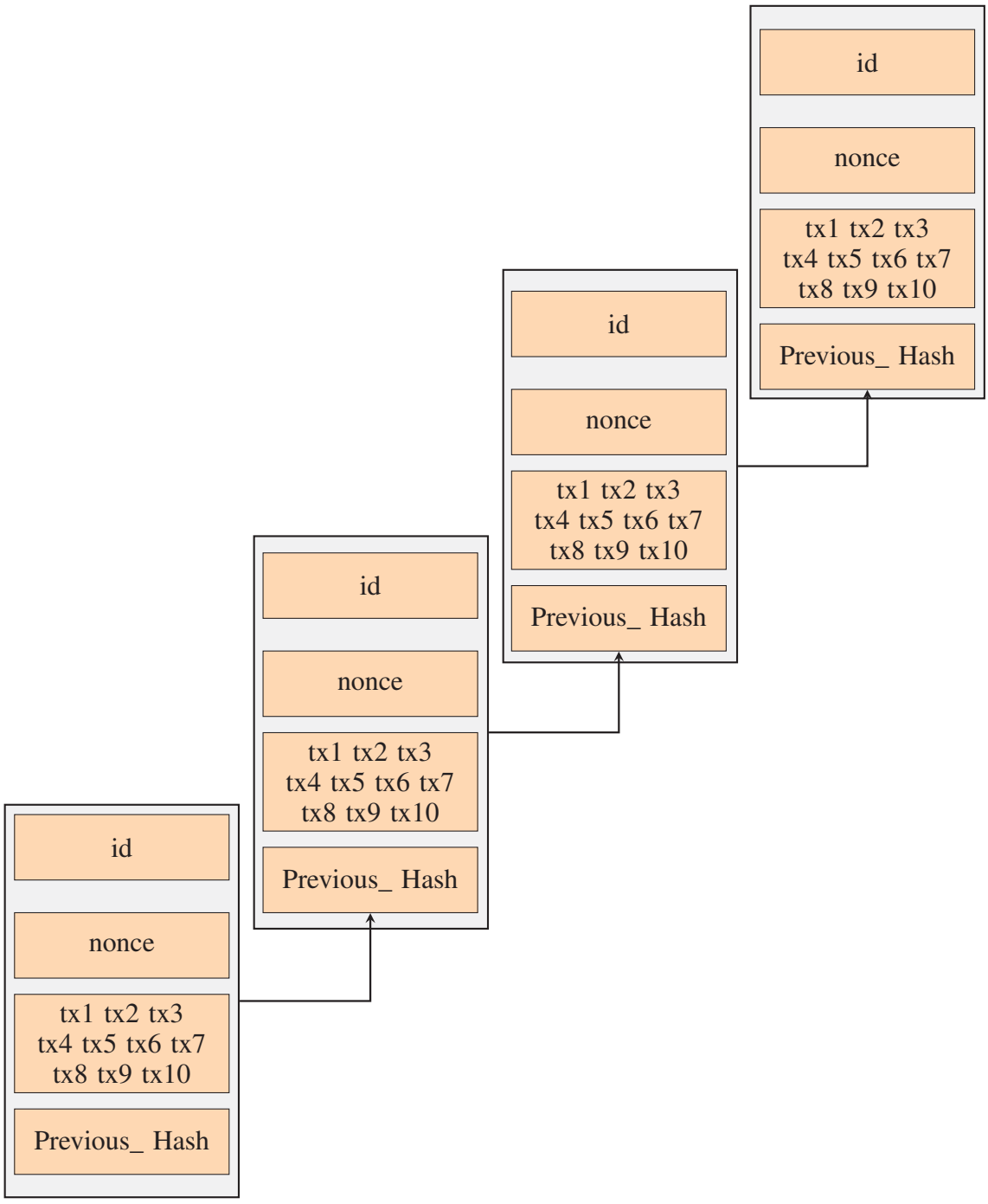

Figure 4.3. Block Visualisation For First Block Chain

For Block-Chain of the gateways, blocks will not be keeping transactions data. Gateways are the servers responsible for updating the devices. They need to be aligned with intended devices all the time, when its needed to send an update, hence gateway mining operations must not be taking resources of the gateway servers. So the block chain of gateways will be keeping only one hash values regarding multiple transactions. This hash values, generated by the gateways, are the leaves and the root of the Merkle tree of specified number of transactions waiting in the queue.

Merkle tree operations are used for the integrity and searching for multiple records. Tracing the tree from root value toward the leaves one can verify the integrity of the leaf values. Here hash of each transaction will form a leaf. The records of all transaction hashes of a root will be stored in an environment like cloud storage or any similar database. But the hashes forming the tree will be placed inside the blocks. Hence the whole transaction information will not be found inside the blocks like its done in the first block chain which is a blockchain for software producers. The chain structure for the block chain of the gateways can be seen in figure 4.5.

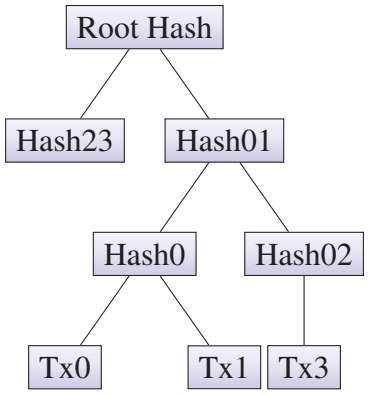

Figure 4.4. Merkle Tree Operation

In (Nakamoto et al., 2008) eason for Merkle tree operation is summarized as follows:

Block headers are about 80kb of data and in a year this will cause 4.2 Mb data to be accumulated if we think that every 10 minutes there is a block mined. He is well reasoned this with Moore?s Law that is predicting 1.2 Gb of growth per year for a memory, storing this block headers will create problem if we do not include actual transaction data inside

the blocks. In our project we think these updating gateway servers as busy machines setting many interaction with the devices. So, Merkle tree solution is well suited for our aim.

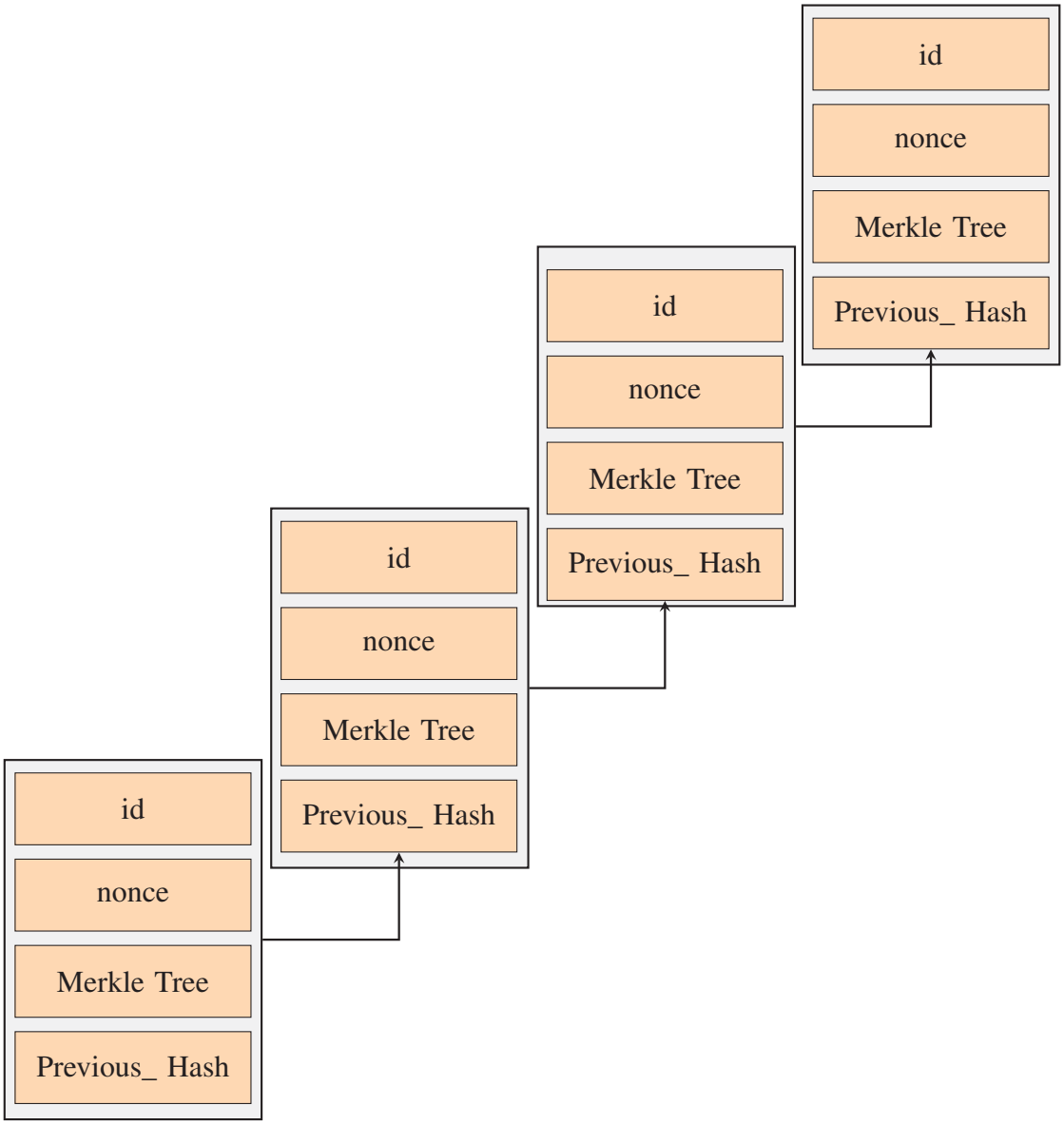

Figure 4.5. Block Visualisation For Gateway's Block Chain

4.1.4. Update

In the update operation, Gateways communicate with the devices on site. Today most of the studies use wi-fi and mobile networks adding extra mechanisms to have secure sender and receiver in order to realize this. An approach where distributers and responders have the roles to verify the operation and repudiate the other side may also be used. But since this study does not cover the distribution of the software through the devices or vehicles these are out of the scope for our study.

4.2. Scenario

In this section the sequence diagram of proposed model is given. Figure 5.7 which shows the general idea of the execution sequence.

Very first operation starts with the OEM company who is responsible for producing software for its devices, either by their own or having support from software producer companies.

The producer wants to keep the newly produced software file in a secure place as well as conveying it to thedevices which it produced for, while keeping its integrity. Then the company will put those in a secure platform like IPFS and forms a transaction which keeps the hash of the file and respective timestamp. It is now authorized since its signed with the private key of the producer. The algorithm for this process is shown in Algorithm 1.

Algorithm 1 Algorithm for Sender

Input: Send_ (<SenderId, ImageHash, DeviceData>)
if (SenderId) valid, (ImageHash) valid, (DeviceData) valid **then**
$build\_transaction$
$Transaction\{SenderId, ImageHash, DeviceData, timestamp, public\_key, signature, id\}$
$Broadcast\ \ transaction\ \ to\ \ all\ \ BC\ \ network$
else
$discard\ \ transaction$
end if

The transaction goes through a network of computers which are responsible for

keeping immutable records of the product features. And this network provides a robust mechanism towards availability attacks since it is a decentralized network of computers, each one is keeping the same chain of information. And this chain is verified using Proof Of Work calculations. The algorithm for this process is shown in Algorithm 2.

Algorithm 2 Algorithm for Miners of the OEMs

Ensure: Listen_Transactions (<SenderId, ImageHash, DeviceData, timestamp, public_key, signature, id>)
Ensure: Listen_ Blockchain_State_Info (<NodeIP, BlockchainLenght>)
Ensure: Send_ Blockchain_State_Info (<NodeIP, BlockchainLenght>)

```
Put Listened Transaction into queue
Get one element from queue
if element is valid then
    continue
    if element is not already in the unmined block then
        continue
        if element is not already in one the mined blocks then
            put this element inside the new block
        else
            quit
        end if
    else
        quit
    end if
else
    quit
end if
nonce = 0
while hash(nonce, block_ hash, valid transactions)
has < 0 s in the beginning do
    nonce = nonce + 1
    if hash(nonce, block_ hash, valid transactions)
has >= 0 s in the beginning then
        add this block to the chaim
    end if
end while
```

As well as keeping the produced software unchanged and authorized there must be also gateways to send those products to the users. Those gateways must be informed the unchanged version of the software. Hence gateways can read the chain of records which is kept by the first network and they can send them to the devices which they are communicating on site. As a result of this communication they can determine the version of device?s software and by reading the chain of records they can detect if there is a

software update for any of the devices on site. Then gateways can ask the actual update files form IPFS via its hashes and send them through to the devices.

After sending updates for devices which need them, gateways must be keeping the records of the updates. But this time again, these records must be protected under integrity requirement to provide liability proof. Again, these network of devices, gateways, will be keeping their update transactions in chain of block as well as sending the update transactions whenever they performed an update for a device. However, since the number of necessary updates is quite high compared to the software records, it is not logical to keep all transaction information in the blocks but keeping a structure like Merkle tree which is preserving integrity is a more viable approach. So a number of transactions will form a Merkle tree now and the hashes of the tree will be kept in the blocks. The algorithm for gateway operations is shown in Algorithm 3.

Figure 4.6 shows whole model working with a sequence diagram. Here first users are the OEM company staff who are responsible for the delivery of the updates. IPFS and the block chain networks are the networks which preserve data with networks? own operations and without requiring third party. Lastly, gateways are another network which is operated on its own, because gateways will be querying for new updates for specific devices. Those servers will be creating the transactions which are outcomes of their own operations. After sending the update a gateway will put its address to as a sender and it will authorize the transaction with its own private key.

Algorithm 3 Algorithm for Miners of the Gateways

Ensure: Listen_ Transactions (<SenderId, ImageHash, DeviceData, timestamp, public_key, signature, id>)
Ensure: Listen_ Blockchain_State_Info (<NodeIP, BlockchainLenght>)
Ensure: Listen_ Devices ()
Ensure: Send_ Blockchain_State_Info (<NodeIP, BlockchainLenght>)
Ensure: Send_ Broadcast_As_Gateway (<NodeIP>)
if there is an update for connected device **then**
Check the version
Receive image file from IPFS and send device
Send an update transaction
end if
Put listened transaction into queue
Get one element from queue
if element is valid **then**
continue
if element is not already in the unmined block **then**
continue
if element is not already in one of the mined blocks **then**
continue
if same update for same device is not already done **then**
put this element inside a merkle tree
put hashes of the merkle tree inside a the new block
else
quit
end if
end if
else
quit
end if
else
quit
end if
$nonce = 0$
while hash(nonce, block_ hash, valid transactions)
has < 0 s in the beginning **do**
$nonce = nonce + 1$
if hash(nonce, block_ hash, valid transactions)
has >= 0 s in the beginning **then**
add this block to the chaim
end if
end while

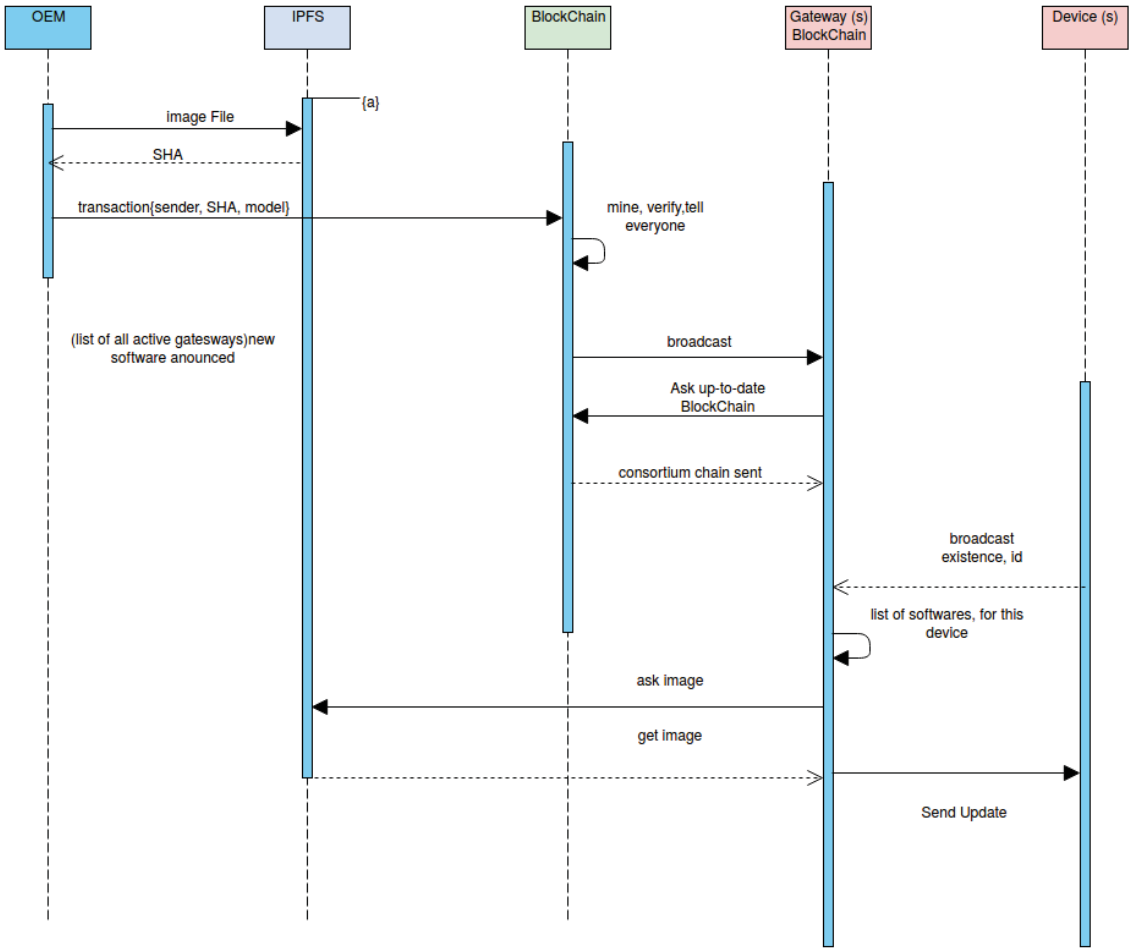

Figure 4.6. Sequence Diagram

CHAPTER 5

IMPLEMENTATION AND EVALUATION

Block chain networks are classified as private, public and consortium networks. As the naming may indicate private block chain networks are special to one organization as they provide solutions for enterprises or business areas. Public block chains are open to anybody who wants to join as one of the nodes in the open network. Last, the consortium block chains are the network of nodes which belong to various organizations and user groups. In order to have one of the nodes there an organization should be a member of a group which shares a common aim to have this chain.

5.1. Implementation

In this part we will provide more details about the implemented the prototype. In this work there are two block chains implemented. Both block chain networks have features of the consortium networks. But for clarity in our model the block chain which is used by the OEMs is named as consortium block chain network whereas the block chain network which is used by the gateways is named as the gateway block chain network. This work is implemented in Eclipse IDE and with Python Socket programming API and run in Python3.6 interpreter. All data is shared in Json format between the members of the block-chain network.

Block Chain Client: Whenever a producer joins the network this producer will have RSA public/private key pairs and respective network address. We already noted that whenever sender sends an update this will be turned to transaction with which is signed with the private key of the sender. The transaction will have an expression with signature and respective timestamp. This is given in Agorithm 4.

$$S = TransacitonDetailsxSenderPrivateKey$$
$$S =< (SenderID, Imghash, modelNO) > xSenderPrivateKey$$
$$Tr =< (S, Timestamp, T_{\text{id}}) >$$

where T_{id}: Transaction id

Algorithm 4 Transaction Handling

Input: (<SenderId>)
Input: (<ImageHash>)
Input: (<DeviceData>)
$signature(S) = PrivateKey(ofSenderAddress)x$
$(sender\_address, ipfs\_document\_hash, DeviceData, Timestamp)$
Output: $TransactionSenderId,\ ImageHash,\ DeviceData,\ timestamp,$
$public\_key,\ signature,\ id$

Consortium Block Chain:

In algorithm 1 it is shown that every transaction will be broadcasted to the network after it is created. After this broadcast operation the transaction will be known with its sender and its verifiable when its added inside a block. Consortium block chain will have the following operations:

- Transaction Handler : Transaction Handler is responsible for listening all transactions broadcasted to the network.
- Network Broadcaster : Network Broadcaster is responsible for broadcasting the chain length to the block chain network.
- Status Handler : Status Handler is responsible for requesting new blocks mined and chained one after another in any other miner's chain data. Status Handler is managing to gather the blocks which are not in this miner's chain. Then it starts adding the missing ones.
- Chain Synchronization : Chain Synchronization stands for synchronizing any miner's chain with another miner's which has shorter block chain than this one. Basically it will serve for requested blocks.
- Network Listener : Network Listener service will be listening the broadcast messages from other nodes. After listening for the updates from other nodes Chain Synchronization service is decided to be called.
- Miner : Miner will be responsible for mining the blocks. As the mine operation requires miner will try nonces starting with '0'. After a brute force sequence of trials miner will find the correct block elements.

These operations will be running in parallel when a node is up and running. Transaction Handler listens to the transaction port and whenever a transaction is received it receives all data with UDP data packets. Then server puts it inside a queue. The queue will

have transactions waiting to be mined. Miner will get one item from queue and do the necessary checks. After checking the transaction, it will continue to put others in or-der to reach the necessary transaction count that must be in a block for this block-chain network. After it reaches the block size defined for this network, the miner will be able to start the mining process. Implementation of the services are adapted from (Codebox, 2018) which is a minimum viable blockchain implementation for a wallet of cryptocurrency.

The necessary checks before selecting a particular transaction includes:

- Checking for signature: Since the public key of the sender and the signed version of the transaction is inside the transaction data, server first inspect the signature performing signing operation by itself.
- Checking if the transaction exist: Since the same transaction must not exsist two times inside a block miner will check the block which is packet that is prepered to be mined at that time. Miner will also check its existence in the block chain.
- Checking the existence of the same update document for the same kind of devices: Miner will be checking if the same upload for the same device family is already existing in both the block to be mined and the whole chain of records.

These steps are given in algortihm 5. The operations given here will be done for every transaction about to be put in a block.

Algorithm 5 Pre-mine checks for consortium block-chain

Input: Transaction(<SenderId, ImageHash, DeviceData, timestamp, public_ key, signature, id>)
if Transaction signature is valid **then**
continue
end if
if This block already does not already has this transaction **then**
continue
end if
if This transaction does not already in block-chain **then**
continue
end if
if A Transaction for the same update file does not alredy in the block-chain **then**
continue
end if
add this transaction tothe current block
Output: *Transaction put into the queue and necessary transaction size decreased one*

Mining operation will continue as long as there is not a valid chain listened by Status Handler and asked from the Chain Synchronization service. If a node in the network have a chain length which is greater than the length of this chain of blocks then the node will request the longer chain from the respective node. It will be asking for that block which itself does not own. In order to have a valid chain each block will be keeping the hash of the block before itself. Hence asked block must be in correct order before they are added to the requester's chain. This operation is shown in Algorithm 6.

Algorithm 6 Miner work for consortium block-chain

Input: predefined number of transactions
  $nonce = 0$
  if hash of the obtained block has the predefined number of zeros **then**
    $block\ is\ mined\ with\ 0\ nonce$
  else
    while hash of the obtained block does not has predefined number of zeros **do**
      $nonce = nonce + 1$
      $calculatehash$
    end while
  end if
Output: $Blockismined$

Gateway Network:

As mentioned previously, network of computers responsible for sending updates for the devices is called Gateway Network for this implementation. Similar with the consortium network of the producer block chain which is a Consortium Network, Gateway Network also in the class of consortium network. In order to differentiate them based on the functionality of the servers we called it as Gateway Network. Gateway Network will have both similar and different kind of operations with the Consortium network.

- Network Listener : Network Listener operation will be listening the broadcast messages from other nodes. After listening for the updates from other nodes Chain Synchronization service is decided to be called.
- Transaction Handler : Transaction Handler is responsible for listening all transactions broadcasted to the network.
- Network Broadcaster : Network Broadcaster is responsible for broadcasting the chain length to the block chain network.

- Status Handler : Status Handler is responsible for requesting new blocks mined and chained one after another in any other miner's chain data. Status Handler is managing to gather the blocks which are not in this miner's chain. Then it starts adding the missing ones.

- Gateway Listener : Gateway Listener will be listening the consortium block chain status information. Gateway Listener will be the trigger for the chain synchronization operation with the block chain network of the producers. Learning what producers implemented new for devices will be the primer cause of the update operation.

- Device Listener : Device Listener is responsible for listening the status information coming from the devices. Devices will be sending their device information given by their producers and the software version they carry at that time.

- Chain Synchronization : Chain Synchronization stands for synchronizing any miner?s chain with the other miner?s which has shorter block chain than this one. Basically, it will serve for requested blocks.

- Miner : Miner will be responsible for mining the blocks. The mining operation requires miner to try nonces starting with ?0?. After a brute force sequence of trials miner will have a block which contains a block number, previous block number in this chain, hashes of a Merkle three and a correct nonce.

Gateway block-chain network will behave in different ways than the consortium network. Besides the services it has common with the consortium network it will also have additional functionalities and additional services. Gateway will be listening the broadcasted status information from consortium network. Since Gateway network will be reading the chain of the consortium network, nodes will be updating their record of chain, which is consortium block chain, length according to the consortium network. This chain of data stands for just reading and being informed that there are new updates for some devices.

Algorithm 7 Gateway Listener

Input: Listen Consortium network broadcast information.
if nodes from consortium chain have block chain lenght longer than we read before **then**
$ask\ for\ the\ newly\ mined\ blocks$
end if

The idea is to remain updated all the time with the consortium network. The process is given in Algorithm 7 for Gateway Listener. On the other hand the actual gateway operations start listening the device information on site and checking for the new updates for related devices from all the immutable ledger of updates. Devices will be identified by their device_no parameter. This number will cover respective brand and model or any other necessary information as well as the number specific to that device. Device identification is given in section 4.1. Whenever a gateway communicates with a device it will list all updates for this device type, regardless of the exact device number. This is for inspecting the last update for this device type. Comparing the software version which is on the device gateway will determine if this device needs update or not. If the device needs an update, gateway fetches the update file from IPFS and send it to the device. In our study gateway and device communication for sending update is not implemented. The implementation work covers all operations regarding blockchains.

After the gateway sends an update to the device it will send a transaction for this update operation in order to be mined as its shown in Algorithm 8. But this time gateway mining will be different than the consortium network.

Algorithm 8 Gateway Operation

Input: have consortum chain readible
Input: have status info messages from devices m(<device_ no, software_ version>)
$list\ all\ updates\ historically\ for\ this\ device\ type$
if device software version is not the has update for this device type **then**
$Connect\ ipfs\ to\ get\ last\ update\ file\ for\ this\ device$
$Send\ last\ update\ to\ this\ device$
$Send\_Transaction(sender\_gateway\_address, hash\_of\_the\_update\_file, complete\_device\_id)$
end if

Transaction sending will be the same procedure as the consortium network. Gateway mining pre-checks and mining operation will be as follows: Before adding a respective transaction to the unmined block, miner will be assured that the current un-mined

block does not have the same update file for the same device and the gateway block chain also does not have the same update file for the same device. It performs the necessary checks like in the consortium network which were checking for the transaction if is duplicated in the current unmined block or in a block in the chain of blocks.

- Checking for signature: Server first inspect the signature performing signing operation by itself. But this time the sender will be a gateway and the transaction to be signed by the private key of that gateway.
- Checking if the transaction exists : Miner will check its existence in the block chain and the current block.
- Checking if the same update document for the same device exists: Miner will be checking if the same upload for the same device already exists in both the block to be mined and the whole chain of records. Also, we do not want the same device to be updated with the same version of software more than once. But our study does not cover this condition since there may be more than one reason for such a case.

These inspections are given in Algorithm 10. The operations given here will be performed for every transaction to be put in a block.

Algorithm 9 Gateway Miner Pre-checks

Input: Transaction(<SenderId, ImageHash, DeviceData, timestamp, public_ key, signature, id>)
if Transaction signature is valid **then**
continue
end if
if This block already does not already has this transaction **then**
continue
end if
if This transaction does not already in block-chain **then**
continue
end if
if A Transaction for the same update file for same device does not alredy in the block-chain **then**
continue
end if
if A Transaction for the same update file for same device does not alredy in this unmined block **then**
continue
end if
add this transaction tothe list of leaf for current merkle tree
Output: *Transaction put into the merkle tree and necessary transaction size decreased one*

All implementations is done in Python programming language with the help of Socket programming. Json format is used for the share of data.

5.2. Evaluation

This study represents an example platform to keep and send updates with availability, authenticity, integrity features. In this work we wanted to show the feasibility of using block-chain networks for such use cases with different regulations. For example, having Merkle tree implementation in the gateway block chain network secured us from having heavy data in the blocks. The reason behind this adjustment which is different than the consortium block-chain network is that gateway servers stands for being ready to communicate with lots of devices on site. Hence there will be many more number of transactions than the consortium network has. There are other studies which designs a block chain network for both keeping the update images and sending updates. Having different networks for sending updates rather than the first aim of keeping immutable records saved us from excessive computation overhead on the nodes. Some other designs

proposes to implement block chain without proof-of-work mechanism for the nodes in an area where lots of transactions are handled. But we wanted to have an application specific environment which serves only for update operations without sacrificing proof-of-work. In the proceeding paragraphs of this section we will first give operation logs of the working prototype and evaluate our study comparing it with the Bitcoin block chain network.

Figure 5.1 shows whole operation designed. The transaction numbers indicated in the figure are explained below.

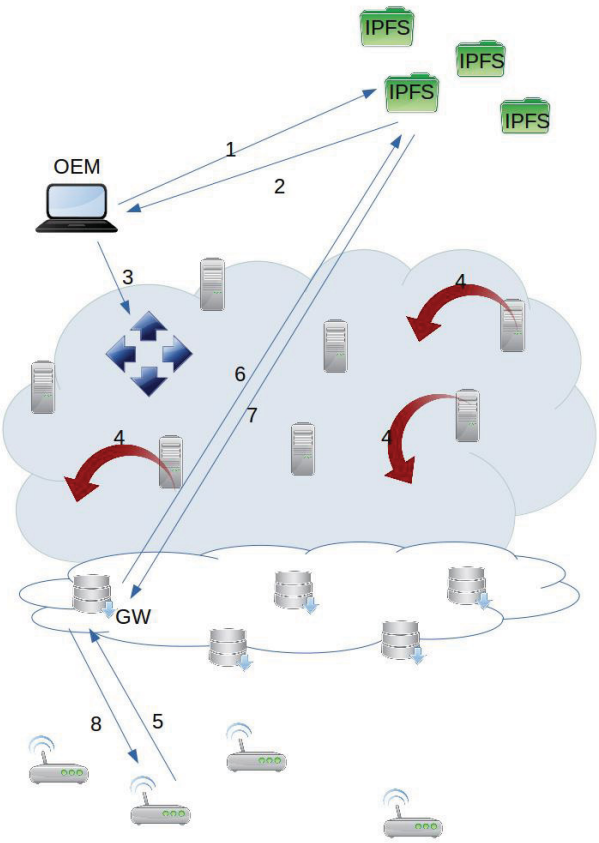

Figure 5.1. Operation

1-2. When the first operation is completed the content addressed internet, that is IPFS, returns the hash of the content of the update file. 3. When the second operation completes there will be transactions broadcasted to the block chain network. Logs in Fig 5.2 shows the situation when the transactions heard from the nodes of the consortium

block chain network.

Figure 5.2. Transaction Log

When the same content for same type of device is broadcasted in a transaction nodes will not add this transaction to their blocks:

Figure 5.3. Transaction Log

4. Mining operation handled by the miner nodes on the block chain network:

Figure 5.4. Mining Log

Gateway block chain network gets updates from consortium block chain network:

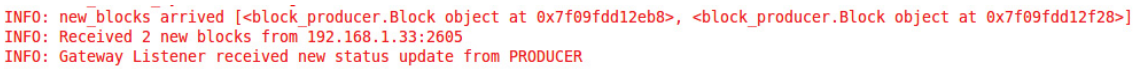

Figure 5.5. Status Updates from Consortium

5. Device listener listens the device informations and checks if there is a new update for those devices.

6-7. If so the gateway node asks ipfs network for the whole content of the update file.

8. Gateway node sends update to the devices and creates an update transaction for its operation. If the transaction is liable it will be mined.

Figure 5.6. Gateway Mining Log

If there is already transaction with same content for the same device it will be ignored.

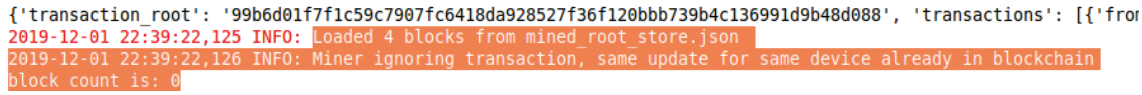

Figure 5.7. Gateway Mining Log2

In this work, besides designing a futuristic remote update model which may cover the lacks of the traditional methods in terms the functionality ee also wanted to have a model which covers the dependency to the third parties and increases impartialness. In the table 5.1, the effects of the 51 percent attack is the risk for appending false blocks. This property of block chains is a possible risk for the liability inspections and making decisions according to the historical record of ledger.

Table 5.1. Comparison of Features with Bitcoin Block Chain

#	Feature	Bitcoin BC	Consortium BC	Gateway BC
1	BC Visibility	Public	Secure/ Private	Secure/ Private
2	Transaction chaining	Input / Output	T are chained to each other/ Output	T are formed a merkle tree/ Output.
3	Transaction mining	All Ts	All Ts [1]	All Ts [2]
4	Mining requirement	POW	POW	POW
5	Forking	Not allowed	Not allowed	Not allowed
6	Double Spending	Prohibited	Not applicable	Not applicable
7	Transaction verification	Signature	Signature	Signature
8	Transaction parameters	input, output, coins.	Block-number, hash of data time, PK	Block-number, hash of data time, PK
9	Transaction dissemination	Broadcast	Broadcast	Broadcast
10	New block verification	Blocks and Ts in blocks	Blocks and Ts in blocks	Blocks and Ts in blocks
11	BC control	No one	No one	No one
12	Miner trust	Miners are all the same.	Miners are all the same.	Miners are all the same.
13	Miner joining overhead	download all blocks in BC	download all blocks in BC	download all blocks in BC
14	Miner rewards	Coins	Nothing	Nothing
15	Pool mining	allowed	Cannot be defined.	Cannot be defined.
16	Malicious miner	Allowed to join	not possible	not possible
17	Effects of 51 % attack	double spending	Increases the possibility of appending false blocks	Increases the possibility of appending false blocks
18	Encryption method	Public/ private keys	Public/ private keys	Public/ private keys
19	Merkle Tree	leafs are in blocks (reclain disk space later)	no merkle tree	leaf transactions are not in the block but in a different database

*For now both of the block chains are configured to mine all transactions. But for a real-world scenario it may be different. For example, if there is one particular brand or model of a vehicle which is used more than others in a region, specific servers may mine transactions belong to those brands in order to tell the gateways that they can distribute the software update for this brand easily instead of lingering with other useless transactions around. Those block chains are capable of this.

Another point is the mining operation of the gateways. The Merkle tree implementation decreases the block sizes to a considerable extent. We saw that Block-chain networks are secure platforms with their resource consuming calculations. We have shown that we could design block chains specific to applications and decrease their resource consumption.

To evaluate our design in terms of security, it is clear that we have open points sending update images from gateways to the devices. But for an attacker model which attacks to the servers of any of the blockchain network, attacker will not able to make every node deny its services. Since we aim to have decentralized networks for both recording the up-date fingerprints and sending transactions we have robust model against denial of service attacks. Another attack model can have the attacker who tries to compromise the chain data. In this case the original design of the block chain prevents this data compromise. This assures that if the attacker does not manage more than half of the nodes of the block chain network, he or she is losing his change to compromise it as time progresses. An attacker model which attacks to the manufacturer must have the private key of it. In another attack scenario which attacks to the network, filtering some of the devices or manufacturer servers to send or receive data, the attacker will be prevented by the built in precautions of the internet like routing tables or predetermined paths. But for an attacker who attacks to a device physically to prevent it to get the update, the attacker will succeed. In another scenario an attacker may want to compromise the information inside update transactions created by the gateway servers. For example, in a case where one vehicle is involved in an accident, depending on the last update the vehicle accrued and the timing for the update of this device is the key information which may be changed. In this case the attacker wants to compromise our future data storage which keeps the transaction information and respective Merkle trees. In this case, since the fingerprints of the transactions are kept inside the block chain, blocks and the hashes of the transactions and the root of the Merkle tree is not the same in the storage field, any of the sides can not claim the compromised timing for the update operation.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1. Conclusion

The main aim of this thesis is to propose a model to send remote software update for embedded devices which are being more dependent on the environment day to day. Especially if there is not a suitable update mechanism to send updates to the devices, with the enhancing functionality of the machines this will create serious problems. That is if there is a future with connected and more capable vehicles, taking the vehicles to the service shops in order to put new software will not be practical. Or with the growing production areas, installing software by hand will not be efficient enough in production. This means that new technologies will be required to handle these operations.

Beside this, in the future connected world, inspections for liability will be more complex because there will be various producers, great number of users and great number of interactions. Case that need to be verified must be leaning on trust based interaction records. Here integrity and authentication are necessary in a trustless environment. Another concern is availability. In order to serve great number of users, servers must be available all the time. Single point failures show us there may be life threatening situations. There must be more viable methods in our lives. What we proposed and designed in this study is a sample model which is suitable for the future use in terms of the requirements we mentioned above. It is a system which involves to blockchain networks which cooperate and perform an update process which helps reduce security issues such as availability and integrity. The proposed solution has been implemented and a feasibility study has been performed. This prototype has to be improved before being applied in a professional production environment since it does not cover the confidentiality area especially in the communication from the gateways to the devices.

6.2. Future Work

Like its mentioned in the conclusion part this model does not cover all the security issues of all interactions. We plan to cover all the interactions in a possible production environment in order to provide a secure model in overall as a future work. Another possible future work may be designing a platform which allows devices to get updates from one another in order to increase the impact of the updates in larger areas.

REFERENCES

Baza, M., M. Nabil, N. Lasla, K. Fidan, M. Mahmoud, and M. Abdallah (2018). Blockchain-based firmware update scheme tailored for autonomous vehicles. *arXiv preprint arXiv:1811.05905*.

Benet, J. (2014). Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*.

Boudguiga, A., N. Bouzerna, L. Granboulan, A. Olivereau, F. Quesnel, A. Roger, and R. Sirdey (2017). Towards better availability and accountability for iot updates by means of a blockchain. In *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pp. 50–58. IEEE.

Codebox (2018). Minimum viable blockchain in python. `https://github.com/codebox/blockchain`. Accessed: 2020-01-04.

David Shepardson, H. S. (2019). Uber not criminally liable in fatal 2018 arizona self-driving crash: prosecutors. `https://www.reuters.com/article/us-uber-crash-autonomous/uber-not-criminally-liable-in-fatal-2018-`arizona-self-/driving-crash-prosecutors-idUSKCN1QM2O8.

Dorri, A., S. S. Kanhere, and R. Jurdak (2016). Blockchain in internet of things: challenges and solutions. *arXiv preprint arXiv:1608.05187*.

Dorri, A., S. S. Kanhere, and R. Jurdak (2017). Towards an optimized blockchain for iot. In *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, pp. 173–178. ACM.

Dorri, A., S. S. Kanhere, R. Jurdak, and P. Gauravaram (2017). Blockchain for iot security and privacy: The case study of a smart home. In *2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops)*, pp. 618–623. IEEE.

Greenberg, A. (2015). After jeep hack, chrysler recalls 1.4m vehicles for bug fix. https://www.wired.com/2015/07/jeep-hack-chrysler-recalls-1-4m-vehicles-bug-fix/. Accessed: 2019-11-06.

Lee, B. and J.-H. Lee (2017). Blockchain-based secure firmware update for embedded devices in an internet of things environment. *The Journal of Supercomputing 73*(3), 1152–1167.

Nakamoto, S. et al. (2008). Bitcoin: A peer-to-peer electronic cash system.

Oham, C., S. S. Kanhere, R. Jurdak, and S. Jha (2018). A blockchain based liability attribution framework for autonomous vehicles. *arXiv preprint arXiv:1802.05050*.

Steger, M., C. Boano, M. Karner, J. Hillebrand, W. Rom, and K. Römer (2016). Secup: secure and efficient wireless software updates for vehicles. In *2016 Euromicro Conference on Digital System Design (DSD)*, pp. 628–636. IEEE.

Steger, M., A. Dorri, S. S. Kanhere, K. Römer, R. Jurdak, and M. Karner (2018). Secure wireless automotive software updates using blockchains: A proof of concept. In *Advanced Microsystems for Automotive Applications 2017*, pp. 137–149. Springer.

Steger, M., M. Karner, J. Hillebrand, W. Rom, C. Boano, and K. Römer (2016). Generic framework enabling secure and efficient automotive wireless sw updates. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–8. IEEE.