2023

The Vulnerability Through Cyberattacks Related to Technological Differences Between Centralized, Server-Based and Ethereum Based Smart Home Systems

Friederike Johanna Haberl
Technological University Dublin, Ireland

The Vulnerability through cyberattacks related to technological differences between centralized, server-based and Ethereum-based Smart Home Systems

Friederike Johanna Haberl

A dissertation submitted in partial fulfilment of the requirements of
Technological University Dublin for the degree of
M.Sc. in Computer Science (Advanced Software Development)

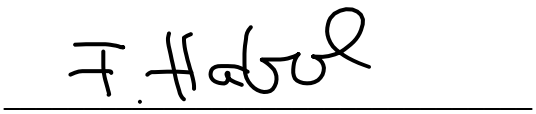
June 2023

DECLARATION

I certify that this dissertation which I now submit for examination for the award of M.Sc. in Computing is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Technological University Dublin and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: 

Date: **17 May 2023**

ABSTRACT

Many people use the digital support of smart home systems to improve their homes' comfort, security, and efficiency (Komninos et al., 2011). However, with the growing number of users comes an increasing number of security challenges that must be addressed (Albany et al., 2022). To mitigate these concerns, the smart home industry is exploring new technologies beyond the traditional centralized server model.

One promising technology is the Ethereum blockchain, a distributed database technology. However, it is unclear which concrete security advantages the use of this technology offers in the smart home area. Therefore, the purpose of this research is to investigate whether there is a correlation between the use of the technology and the security of smart home systems against cyberattacks. To accomplish this, it is examined whether an Ethereum-based smart home system can defend against a larger variety of different cyberattacks than a classic smart home system with a central server.

Key words: *Smart Home, Blockchain, Ethereum, Cyberattacks, Security*

ACKNOWLEDGEMENTS

First, I would like to express my sincere thanks to my supervisor Dr. Bojan Bozic for his help, guidance, and assistance throughout the course of this dissertation, without his unrestricted help I would have struggled to write this thesis.

I would also like to thank Prof. Dr. Frank Zimmermann from the Nordakademie - Hochschule der Wirtschaft, who actively supported me by enabling me to study for a master's degree.

Finally, I would also like to thank my friends and family who supported me not only during my dissertation but also throughout my master's degree. Without you all, I would have never managed to finish these masters without losing my mind.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1 INTRODUCTION

1.1 RESEARCH BACKGROUND

Continuous technological development has led to an improvement in the quality of people's lives (Komninos et al., 2011). One such technology is a so-called Smart Home system which describes the use of technology in a living environment to increase aspects such as comfort, security, and efficiency of the living space. To implement a Smart Home system, the so-called Internet of Things (IoT) is used, which deals with the connection of things via the Internet (Albany et al., 2022). Consequently, a smart home system consists of a group of connected IoT devices that can be controlled remotely via smartphone or laptop.

In recent years, the use of IoT devices and smart home systems has grown significantly, leading to the expansion of this technology into various aspects of life (Albany et al., 2022). However, this expansion entails greater security challenges that need to be overcome. Due to the security risks, there are various architectures and technologies that are used in the Smart Home sector. Among these architectures, there is the classic Smart Home system using a central server (Ahmed et al., 2021) and a novel approach where Smart Home Systems use a special distributed database technology which has been proven to bring security advantages (Albany et al., 2022).

This special technology is called blockchain and describes a distributed database consisting of cryptographically linked blocks (Ozyilmaz & Yurdakul, 2019). Each block contains a set of transactions, which becomes immutable by broadcasting it to the network. One specific type of blockchain is Ethereum, which has been used in IoT infrastructure designs and therefore, can be applied to Smart Home Systems.

1.2 RESEARCH PROBLEM

As the use of smart home systems in private households increases, the importance of security aspects is growing due to the fear of users of unauthorized access to private data through passive and active cyberattacks (Komninos et al., 2011). There are several ways to design a smart home system to increase its security. The classic smart home system involves a server-based approach where a central server component is used to

communicate with the user and to which sensors, like a temperature sensor capturing the current temperature, send information (Ahmed et al., 2021). Another approach uses a blockchain as an authorization or payment method for accessing the underlying smart home system (Benlamri et al., 2020). However, the problem in this approach is that the central server underneath the protective layer of the blockchain is still used as the core component of the system. A new approach is needed to fully utilize the potential of the blockchain in smart home systems.

Since blockchain is a distributed database and server's function in the classic smart home system can be limited to storing data in the database, the blockchain can be used similarly to the server within the smart home system. However, it is crucial to determine whether this new approach can guarantee advantages in the field of security of smart home systems compared to the classic system. To evaluate whether the new approach improves or degrades security, this research focuses on the vulnerability to cyberattacks in the context of technological differences between centralized, server-based and Ethereum-based smart home systems. Consequently, the following research question is created: Can an Ethereum-based smart home system defeat a higher number of different cyberattacks than a centralized, server-based Smart Home system? Further information on the research question is provided in chapter 2.4.

1.3 RESEARCH OBJECTIVES

While other studies focus on improving the security of smart home systems by adding new or additional security mechanisms to a system, the goal of this research is to understand the correlation between the technology used and the vulnerability to cyberattacks for smart home systems. For this purpose, a comparison is made between a centralized, server-based and an Ethereum-based smart home system specifically in terms of their security against cyberattacks.

This research contributes to the development of more secure smart home systems because if a correlation between the security of smart home systems and the used technology can be proven, this would mean for future research that its research fields would change. Because if the core technology of a smart home system has fewer security problems, then this means less effort must be spent on researching additional security mechanisms. This in turn means that future researchers would have to look more at the security of different technologies to create an improved generation of smart

home systems, rather than improving the security of an existing system by reducing the negative effects of the involved technology.

The main objectives of this dissertation are as follows:

1. Investigate the functioning of centralized, server-based smart home systems and their security issues related to cyberattacks.
2. Investigate the functioning of the Ethereum blockchain and its security issues related to cyberattacks.
3. Design two smart home systems, one using a centralized, server-based approach and the other using Ethereum blockchain and evaluate experiments that can lead to answering the research question of whether an Ethereum-based smart home system can defeat a higher number of different cyberattacks than a centralized, server-based smart home system.
4. Create the designed prototype in a digital manner and execute the evaluated experiments against the prototypes.
5. Confirm whether an Ethereum-based smart home system can defeat a higher number of different cyberattacks than a centralized, server-based smart home system by using the result from the experiments.

1.4 RESEARCH METHODOLOGIES

This research consists of primary, empirical research and provides an inductive approach by comparing two designed smart home system prototypes. As with primary research, the used data has been gathered through self-conducted research and the result of this research is measurable which makes the research empirical. This research is quantitative research by testing the relationship between used technology and security issues around smart home systems. Furthermore, the use of empirical evaluation techniques provides an inductive basis for understanding and advancing security-enhanced smart home systems.

The applied research methodologies are visualized in the following diagram.

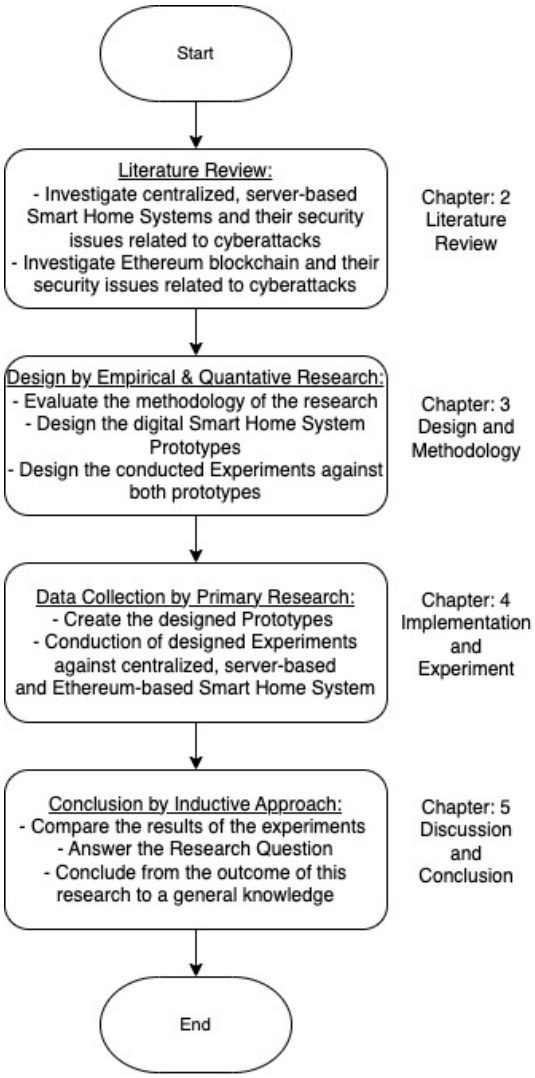


Figure 1 Flow Chart of Research Methodologies

1.5 SCOPE AND LIMITATIONS

The main objective of this dissertation is to investigate security risks posed by cyberattacks in both traditional centralized, server-based smart home systems and Ethereum-based smart home systems. Therefore, two smart home systems are designed and digitally built. In the design, the architecture of the traditional smart home system is used as the basis and is adapted to the Ethereum-based smart home system to ensure the comparability of both systems. In the design of these systems, no additional security mechanisms have been considered and only the minimum for a functional prototype has been implemented, so that only one user, one sensor, one controller, and the respective technology, server or blockchain, have been considered and implemented. For the design, it was assumed that the components of the system

communicate wirelessly with each other, but it was not considered whether this was implemented via the Internet or other methods such as Bluetooth or Zigbee. Furthermore, the design of the systems and the experiments did not consider networks or network security mechanisms, so that obstacles caused by i.e., an implemented firewall, VPN, or simply different network systems are excluded. Furthermore, for the design of the experiments, it was limited to and selected five types of cyberattacks, which are executed against both prototypes equally for 100s. It was assumed that the interfaces of both systems are known to the attacker, since in a real system the attacker can perform a network scan within a network, which would reveal the interfaces of the system to him. But since the two systems are simply digital prototypes and there is no need to look at different networks, knowledge of the interfaces can be assumed.

Furthermore, there are limitations to the research. These limitations include that when creating a digital self-designed centralized, server-based Smart Home prototype and a digital self-designed Ethereum-based Smart Home prototype with the help of the ETH.build application, the smallest differences to a real, non-digital prototype can arise, which result from the specific implementation of the ETH.build software but also from the available functionalities ETH.build offers to the user. Therefore, these changes can influence the outcome of this dissertation.

1.6 DOCUMENT OUTLINE

This dissertation is organized as follows:

Chapter Two covers a literature review regarding Smart Home and Ethereum. Firstly, it is examined what a smart home system is and how a centralized, server-based smart home system works. Afterwards, the security issues by cyberattacks related to a centralized, server-based smart home system are examined. Secondly, the literature review focuses on blockchain technology and its functionality. Furthermore, it described the specific blockchain Ethereum and which security issues by cyberattacks an Ethereum blockchain faces.

Chapter Three concentrates on the methodology of this research and how the research question is going to be answered. Afterwards, a design for a centralized, server-based smart home system prototype and an Ethereum-based smart home system prototype is developed. Lastly, experiments for both

prototypes are designed. Therefore, cyberattacks for these experiments are chosen by using the literature review and the conduction of these experiments are defined.

Chapter Four focuses on the creation of the designed prototypes and the execution of the experiments. The result of the experiments is documented in this chapter.

Chapter Five discusses the result of this dissertation. Furthermore, limitations of this research, as well as ideas for future work, are described in this chapter. Finally, the dissertation is concluded.

2 LITERATURE REVIEW

In the following chapters, two areas of research are targeted that reflect the key areas of concern for this study. Firstly, it is defined what a smart home system is, how the classic smart home system with a central server is functioning, and which security problems it has. Secondly, it is defined what a blockchain is, which specifics Ethereum has, and which security problems exist with Ethereum. Subsequently, the existing literature is critically reviewed, which leads to the formulation of a research question.

2.1 SMART HOME SYSTEMS

A smart home system consists of Internet of Things (IoT) and has the purpose of enabling homes to become smarter and thus safer, more efficient, and more comfortable (Abdullah et al., 2019). IoT refers to the connection of electronic devices and objects that can connect to the Internet and transfer data. For smart home systems, sensors are integrated into residents' devices so that they can be remotely operated or monitored.

There exist various approaches to implementing smart home systems. The classic smart home system uses a central server as a core component (Ahmed et al., 2021). The structure of the classic smart home system is described in the following chapter.

Lots of research has been done on the classic smart home system and therefore, security threats have been identified, which are explained in Chapter 2.1.2.

2.1.1 STRUCTURE OF CLASSIC SMART HOME SYSTEMS

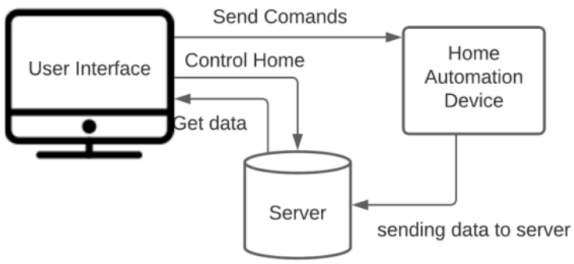


Figure 2 Basic Structure of Classic Smart Home System (Ahmed et al., 2021)

Figure 2 shows the structure of the commonly known smart home as described in the article by Ahmed et al. (2021). The article deals with the design of a smart home

system consisting of a user interface, a server, and home automation devices. The home automation devices or sensors continuously send data to the server. Afterwards, the user can query the server for this data and control the smart home via the server. Furthermore, the user can also send commands directly to the home automation devices as the devices can execute multiple tasks.

The home automation devices are organized so that several microcontrollers exist to which multiple sensors send data at the same time so that the microcontrollers in turn forward the data to a master microcontroller (Ahmed et al., 2021). The master microcontroller sends the collected data to the server or the server's database.

The communication between the sensors and the microcontrollers is wired whereas the communication between the user or user interface and the server is wireless (Ahmed et al., 2021). In the article by Ahmed et al. (2021), it is described that wireless communication takes place via HTTP. The user sends an HTTP request to the server, which responds with an HTTP response.

The described workflow can be seen in Figure 3.

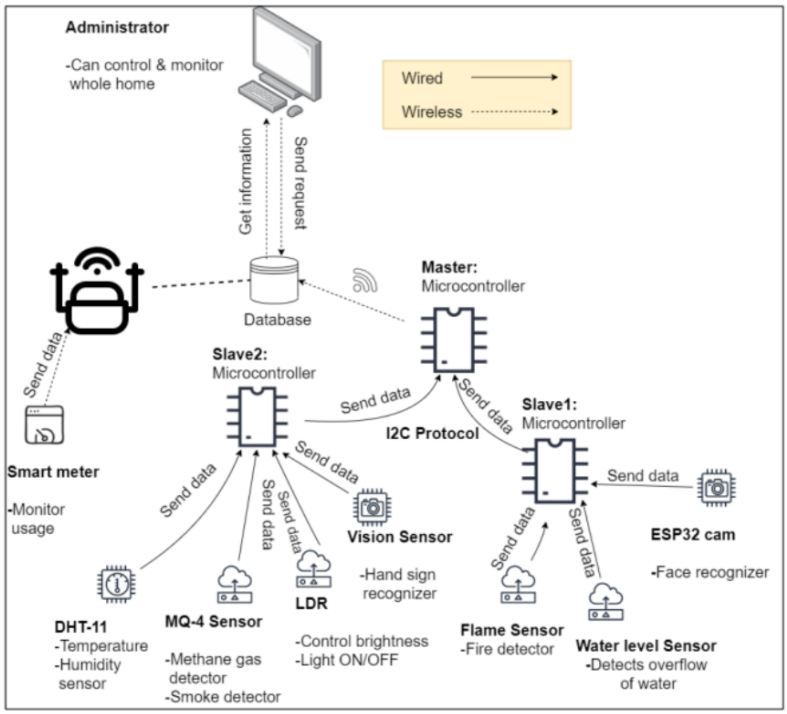


Figure 3 Detailed Structure of Classic Smart Home System (Ahmed et al., 2021)

In addition to the research on the classic smart home system, its security threats have been evaluated and are listed in the next chapter.

2.1.2 SECURITY THREATS OF CLASSIC SMART HOME SYSTEMS

Smart home systems are affected by internal and external threats that can compromise the security of the system (Komninos et al., 2011, p. 184). Internal threats address issues that are triggered by the internals of the smart home system, such as the internal network in which the smart home system resides. These internal threats can be caused by network constructions, misconfigurations, or insufficient security plans and can result in creating security risks for the user and thus holes for intruders.

Furthermore, there are external threats that can affect the security of a smart home system (Komninos et al., 2011, p. 184). External threats are malicious nodes that are located outside the system. These external threats are classified into passive and active attacks. In passive attacks, intruders attempt to gain unauthorized access to transmitted information without modifying it. Since the attacker does not modify the communication, but only observes it, it is difficult to detect such attacks.

Two passive attacks are the so-called Eavesdropping and the Traffic Analysis (Komninos et al., 2011, p. 184). In the eavesdropping attack, the attacker monitors the traffic between a user, the smart home system, and the outside world without the parties noticing (Komninos et al., 2011, p. 185). This captured information could be confidential information that the user does not want to be leaked to unauthorized third parties. Moreover, eavesdropping is classified as the most widely identified security problem in open networks.

Traffic analysis is a passive attack in which the attacker attempts to extract information from the communication of the smart home environment (Komninos et al., 2011, p. 185). This allows the attacker to extract sensitive information such as passwords from the exchanged messages. It is irrelevant whether the messages are plain or encrypted and cannot be decrypted. The more messages the attacker can capture, the more information the attacker can extract from them.

In active attacks, the attacker intends to manipulate information or generate false data and inject it into the internal network of the smart home system (Komninos et al., 2011, p. 185). Such active attacks can result in various losses for the user. There exists the masquerading, replay, message modification, and denial-of-service attacks. In masquerading attacks, the attacker pretends to be a legitimate user or entity of the

system and thus obtains unauthorized privileges. The attacker tries to get sensitive information or access to services through this access.

In a replay attack, an attacker first gains access to messages exchanged between two authorized communication partners and then later retransmits them as an authorized instance. The attacker can copy a valid service request generated from a device within the smart home environment and replay it to gain access to the service that the home user is authorized to access.

Message modification occurs if an attacker alters the content of a message by deleting, adding, or changing it. In addition, during this attack, an attacker may also delay some messages or change their order, resulting in an unauthorized impact. Modification of messages can take place when an attacker has the intention to modify the communication between two legitimate entities in such a way that it maliciously works or modifies information in a data file. Message Modification attack can lead to a DoS attack and represents an attack on integrity.

A denial-of-service attack is used to make the attacked network unavailable or to reduce its availability (Komninos et al., 2011, p. 185). The attacker sends numerous messages to the system to overload or saturate its resources. This restricts the user from using the services of the smart home system. The attacker can attack the server or the devices of the smart home system so that the internal traffic of the system is blocked.

An overview of all mentioned external security issues can be seen in Table 1.

Table 1 Overview of External Security Issues related to Classic Smart Home Systems identified by Komninos et al. (2011, p. 184-185)

Passive Cyberattacks	Active Cyberattacks
- Eavesdropping Attack - Traffic Analysis Attack	- Masquerading Attack - Replay Attack - Message Modification Attack - Denial-of-Service Attack

2.2 BLOCKCHAIN

A blockchain represents a distributed database that is deployed in a peer-to-peer network (Ozyilmaz & Yurdakul, 2019, p. 2). Nodes operate within this network, continuously creating transactions and broadcasting them to the system. The

blockchain consists of individual blocks that contain, amongst other things, the broadcasted transactions and are cryptographically linked to each other. Each transaction in the system is signed using public-key cryptography and can therefore be verified.

A blockchain consists of successive blocks where each block is cryptographically linked to the previous block (Benlamri et al., 2020). A block consists of a block number, the block hash of the previous block, transaction data, a nonce, a timestamp, and the hash of the current block. To create a new block, there are so-called miners in the system, who search for a nonce value. Miners in traditional blockchain systems must find a solution to a difficult-to-calculate but easy-to-prove mathematical problem (Ozyilmaz & Yurdakul, 2019, p. 2). For the problem, there exists a difficulty, which continuously changes and must always be satisfied. The result of the calculation is called a nonce and must be a valid hash value. The mathematical problem could be finding a valid hash value that starts with a specified number of leading zeros (Benlamri et al., 2020). The miner that finds a valid nonce first is declared the winner and is allowed to add the block to the blockchain. Since there can be multiple miners, there are different consensus algorithms that decide which mined block is valid.

In addition to the components of a blockchain already mentioned, there is the ledger (Alex & Stephen, 2018, p. 6). A ledger is a decentralized application that is assigned to each user in a blockchain. After a transaction has been carried out between two users of the blockchain, it is automatically recorded into the ledger. For example, if person A sends 100 money units to person B, then this transaction is recorded in the ledger of all participants of the blockchain. If there is a discrepancy, a voting mechanism is triggered to validate or reject the transaction.

Table 2 Overview of Categories and Types of Blockchains

Category	Types
Blockchain	- Public - Private
Consensus Algorithm	- Proof-of-Work - Proof-of-Stake - Practical Byzantine Fault Tolerance
Cryptography	- Digital Signature with Public-Private Key Pairs
Nodes	- Miner

	- Full Node - Thin Client - Server-trusted Client

An overview of categories and types of blockchains can be seen in Table 2 and are explained in the following paragraphs.

In general, blockchains are distinguished between public and private blockchains (Aung & Tantidham, 2017). Public blockchains mean that anyone can enter the network and thus anyone can participate in the mining process. The unrestricted participation of users is the reason why the use of the consensus algorithm is important to eliminate malicious participants. A private blockchain is defined as one where only one specific owner can trade the blockchain network. Therefore, private blockchains do not need a mining mechanism. To prevent unauthorized users from appending new blocks, smart contract features can be used to define access policies.

There are several types of consensus algorithms. A distinction is made between Proof-of-Work (PoW), Proof-of-Stake (PoS), and Practical Byzantine Fault Tolerance (PBFT) (Ozyilmaz & Yurdakul, 2019, p. 3). In PoW, miners must solve a difficult-to-solve but easy-to-verify mathematical problem. This means that the miner needs high computational power to create a block. PoS is another consensus approach where the creator of the next block is chosen randomly, and the randomness is dependent on the number of coins of the peer. This means that less energy is used for PoS than for PoW, as less computational work needs to take place, but it is more expensive. Finally, there is the PBFT consensus algorithm, which tolerates faults in distributed, low-latency storage organisations by receiving a set of identical responses from the nodes.

As mentioned before, each transaction is signed using public-key cryptography. This is also referred to as digital signatures (Aung & Tantidham, 2017). Each participant in a blockchain network has a public and a private key. The private key is used to sign the digital transactions and must therefore not be passed on. The digital signature involves two different phases. One is the signing phase and the other is the verification phase. For example, User A wants to send a message to User B. Therefore, user A encrypts her data using her private key and sends the result to user B. User B verifies the data by validating it with user A's public key. This allows user B to verify whether the data has been tampered or not.

As mentioned beforehand, the types of nodes in the system may also differ. There are the miners, which pack transactions into a block and run consensus algorithms to meet the requirements of the system to get a financial benefit from it (Ozyilmaz & Yurdakul, 2019, p. 3). Besides the miners, there are full nodes, which have downloaded the entire blockchain and thus continuously verify the integrity of all transactions. Then there are thin clients, which only download the block headers with the hash of the previous and current block to save storage and computing resources. Finally, there are server-trusting clients, which can be used to connect to a server and execute queries against the blockchain.

While blockchain is only a concept of the technology, Ethereum is an actual existing implementation whose features are described in the following chapter.

2.2.1 ETHEREUM

Ethereum is a type of blockchain with decentralized features that lead to developers being able to build and deploy decentralized applications (Alex & Stephen, 2018, p. 6). Ethereum is a distributed public blockchain network, which means that the network needs the current state of information for each Ethereum application. This includes the balances of the users, the code of the Smart code contracts, and the storage location. Smart Contracts are computer programs that directly control the transfer of digital currency. These contracts are stored directly on the blockchain and thus also form a decentralized system.

In summary, Ethereum has the character of a public blockchain (Alex & Stephen, 2018, p. 6), uses Proof of Work as a consensus algorithm (Ozyilmaz & Yurdakul, 2019, p. 3) and uses Digital Signatures as a cryptography algorithm since Ethereum is a blockchain.

Ethereum-related security threats are identified and listed in the following chapter.

2.2.2 SECURITY THREATS OF ETHEREUM

The concept of blockchain, and therefore the Ethereum blockchain, is affected by certain security issues. There is the distributed denial-of-service attack (Alex & Stephen, 2018, p. 3). This attack aims to slow down, shut down, or crash the target system. This can be a significant business risk, especially for blockchains, and is particularly impossible to prevent. In addition to the high risk of a distributed denial-

of-service attack, there is also the risk of a man-in-the-middle attack. This attack is also known as third-party interaction because the attacker tries to get between a user and the blockchain and thereby access sensitive data or modify messages.
An overview of all mentioned external security issues can be seen in Table 3.

Table 3 Overview of External Security Issues related to Ethereum identified by Alex & Stephen (2018, p. 3)

Passive Cyberattacks	Active Cyberattacks
- None	- Distributed Denial-of-Service Attack - Man-in-the-Middle Attack

2.3 CRITICAL EVALUATION OF LITERATURE

The articles by Ahmed et al. (2021), Feher et al. (2022), Sisavath and Yu (2021), Gunawan et al. (2017), and Hyeong-Ah et al. (2014) focus on the design of traditional smart home systems and the evaluation of the created design. All articles place different focuses on their developments, resulting in different evaluations and thus different security vulnerabilities. While Ahmed et al. (2021), Feher et al. (2022), and Sisavath and Yu (2021) limit themselves to certain types of sensors when creating a Smart Home prototype, but their prototypes are connected to the Internet, the elaboration by Ali and Awad (2018) does not examine sensors, but it does include a connection of the smart home system to the Internet. In contrast, the prototype of Gunawan et al. (2017) and the elaboration of Arabo (2015) are not limited to specific sensors, but they are not connected to the Internet. The connection of the system to the Internet alone results in many passive and active possibilities for attacks, which were elaborated by the article by Mantas et al. (2010). Their article deals exclusively with the security aspects of a Smart Home environment. However, this article also has gaps, as security vulnerabilities due to the sensors used were not considered. The article by Hyeong-Ah et al. (2014) again brings other aspects into the security risks of a smart home system by referring to different types of applications, devices, operating systems, and even communication protocols, which however considers the security factor only in the focus of the communication protocols, so that different security risks are mentioned compared to the other articles. The article from Sisavath and Yu (2021) is the only article that brings out the connection between Smart Homes and Internet of Things technology but lacks in elaboration a secure system to prove that its design is

secured against passive and active attacks. Furthermore, the article by Hmeidi et al. (2019) has a completely different approach, so its security risks are based on the architectonics of the smart home system and consider its risks per layer without including specifications of individual layers. Abdullah et al. (2019) and Albany et al. (2022) also deal with the security risks of smart home systems, but only take a very rudimentary approach in terms of detail, so that different types of sensors or communication paths are not considered. The article by Albany et al. (2022) describes various systems that are intended to reduce security risks but does not mention how these systems do so. The same applies to the elaboration of Kim and Robles (2010), who describe, among other things, that the integration of a blockchain would be useful for the security of the system, but hardly substantiates this claim. Lastly, the article by McAuley et al. (2021) is a single article that includes the security risks posed by technology and humans and analyses them. The article mainly focuses on standards that should be used and therefore this article is more theoretical in nature as these standards have not been implemented in any practical way. In summary, the above articles all deal with the smart home system and its security issues. However, each of these articles focuses their research on different specifications of the smart home system, such as the different types of communication between the components of the system or the structure of the smart home system and argues for or against the occurrence of security problems based on this. None of these articles, though, addresses the core of the smart home system, the technology used. This seems to be untouchable for all the articles so only research around the technology was done.

While the previously mentioned articles deal exclusively with Smart Home topics, the articles by Alphand et al. (2018), Dorri et al. (2019), Benlamri et al. (2020), Abbas et al. (2022), and Ozyilmaz and Yurdakul (2019) deal with Internet of Things systems that use blockchain. The articles by Benlamri et al. (2020) and Abbas et al. (2022) contain information to develop a blockchain-based smart home system. The article by Alphand et al. (2018) also proposes a blockchain architecture that has security and can prove it with experiments. However, the research is not based exclusively on one specific blockchain, but on blockchains in general. But an Ethereum-based system was used for the experiments, so emerging security risks from using Ethereum were not considered. The article by Dorri et al. (2019) brings out serious security risks of the Internet of Things systems using blockchain that should be considered when creating such a system. Nevertheless, the research only looked at public blockchains, so a

potential Internet of Things system with a private blockchain does not necessarily have to have such security risks. The last articles considered deal with the integration of Blockchain in IoT systems and thus bring together the new technology with an IoT system. Unfortunately, there seem to be gaps in their research, so that for example the research by Alphand et al. (2018) is based on the general concepts of blockchain, but their experiments use a concrete blockchain, which only applies certain concepts of a blockchain and thus may differ from the theory of blockchain in details. It should also be noted that while a smart home system uses the Internet of Things, not every Internet of Things system is a smart home system. This means that there is still another step missing in the research towards smart home systems and new technologies.

The article by Alex and Stephen (2018) deals exclusively with the security of blockchains and points out security problems through denial-of-service or man-in-the-middle attacks, which could not be prevented by any security mechanisms of a blockchain. This article can be relevant for future research about the security of blockchains and smart home systems, but it only covers blockchains in general instead of one specific blockchain like Ethereum. But there was no article found about the security issues of Ethereum, so this research needs to be done in the future.

Lastly, the articles by Ozyilmaz and Yurdakul (2019) and Patruni and Saraswathi (2022) are the only articles that bring the Internet of Things together with the specific blockchain Ethereum to design a blockchain-based Internet of Things system. However, the article by Ozyilmaz and Yurdakul (2019) leaves out which security risks may arise from this design while the article by Patruni and Saraswathi (2022) deals with security aspects but leave out security risks that other articles mention. For example, the article only deals with denial-of-service attacks, whereas the article by Alex and Stephen (2018) also mentions the security risks of man-in-the-middle attacks.

There is a gap in the research of smart home systems and the use of new technologies like the Ethereum blockchain. Articles that deal exclusively with smart home systems and their security problems take the technology used for granted and therefore only research which security problems arise due to specifics surrounding the technology, such as the design of the system or the types of communication. Other articles deal with the interaction of blockchains in general, but also Ethereum in an IoT system, but only partially address the resulting security issues and even its results are only valid in part for the smart home area, because not every IoT system is a smart home system. From these results, it is obvious that there is a need for research on security issues

related to the technologies used in the smart home area. Specifically, the research needs to look at a specific blockchain and see to what extent the technology used has an impact on the security of a smart home system.

This means that the following points must be considered during this research:

1. Specific technologies must be compared, not just their theoretical concepts.
2. The technologies need to be put in context with concrete smart home systems rather than just with IoT systems.
3. It must be clear which security problems arise from the use of technology in the smart home system or alternatively where these differ.

2.4 RESEARCH QUESTION

Most articles deal with the security of smart home systems by analysing or improving existing systems. The problem is that the existing systems are viewed as a whole and not their underlying concepts or technologies. This means that these articles only change the impact of the system, but not its core problem, the technology and used architecture, which causes high attackability and thus poor security of smart home systems. It has been proven that Ethereum blockchains can ensure higher security due to their structure and concept and are less vulnerable to cyberattacks compared to a classic smart home system using a centralized server.

A few articles have examined the extent to which blockchain can be used in the smart home sector and have used Ethereum. It has been proven to have a high level of security. However, the issue is that these two systems are not comparable, as the articles have changed not only the underlying technology but also the entire concept of using the system. Since the comparability is missing, it is not measurable to what extent an Ethereum-based smart home system is more secure than its classic server-based smart home system competitor.

Furthermore, different articles are based on different aspects of security. Some articles use internal threats and some use external threats to estimate the existing security level of a system. Thus, differences exist not only in the structure of the systems but in the assessment of security. Therefore, this must also be made comparable by assessing the security level of a system by defending against external threats or cyberattacks in this research.

Consequently, the following research question arises: Can an Ethereum-based smart home system defeat a higher number of different cyberattacks than a centralized, server-based Smart Home system?

This research question closes the gaps by basing both systems on the same concepts and architectures with the only difference being the used technology. By testing two equal systems, comparability of both systems can be established to get a meaningful answer on which system can guarantee better security when executing similar cyberattacks in the same quantity.

To close this gap and address the security question, it is necessary to create a possibility to use two comparable smart home systems. Since there are no existing comparable systems, two comparable systems must be created in this research. To do this, digital tools can be used. There is an application called ETH.build, which is capable of building and using a classic and Ethereum based smart home system in digital form. By using this application, the following hypothesis can be formulated: If an ETH.build self-designed digital Ethereum-based Smart Home System prototype is used, then the number of defeated different cyber-attacks is increased compared to an ETH.build self-designed digital centralized server-based Smart Home System prototype. Accordingly, the null hypothesis is formulated as follows: If an ETH.build self-designed digital Ethereum-based Smart Home System prototype is used, then the number of defeated different cyber-attacks is not increased compared to an ETH.build self-designed digital centralized server-based Smart Home System prototype.

3 DESIGN AND METHODOLOGY

This chapter explains the methodologies used to implement this research and how they were selected. Subsequently. two smart home system prototypes, using the existing literature, are designed, one of which corresponds to the central server approach or the classic smart home system and the other to a new type of approach with the Ethereum blockchain as a core component. Afterwards, experiments against both prototypes are designed to evaluate the difference in defeating cyberattacks between both systems using different core technologies.

3.1 METHODOLOGY

The core question of this research is whether a smart home system with a central server can defeat more different types of cyberattacks than a smart home system with Ethereum as core component. The focus of this question is which of the two technologies can defend themselves better without additional security mechanisms. Therefore, when designing both systems, it is important that they are comparable and do not differ in structure and functionality except for the core technology used. Furthermore, it is important, that both smart home systems do not implement additional security features such as authentication to ensure focused testing of the technology. When this is ensured, then it is also ensured that the possible different results of the experiments are not related to other differences in the systems but solely due to the technologies used, server, or Ethereum blockchain.

For the results of the experiments to be expressive, the same cyberattacks are carried out against both prototypes and the results are documented. An attack is considered successful if its definition is met, for example in the article by Komninos et al. (2011, p.185) it is described that an eavesdropping attack aims to monitor the traffic between users, smart home systems, and the outside world without the parties being aware of it. If an attack on a system can monitor only parts of the communication without the parties being notified, then the attack is already considered successful.

The research question can be answered by comparing the amount of successfully defeated cyberattacks of both prototypes. If the Ethereum-based smart home system prototype has more successfully defeated cyberattacks than the centralized, server-based smart home system prototype, then the research question can be answered with

yes. The Ethereum-based smart home system can defend a higher amount of different cyberattacks than the centralized, server-based smart home system. However, if the Ethereum-based prototype has the same or fewer amount of defended cyberattacks than the server-based smart home system prototype, then the research question is answered with no. The Ethereum-based smart home system cannot defend against a higher amount of different cyberattacks than the centralized, server-based smart home system.

Since there are no existing smart home systems that use a central server for one and an Ethereum blockchain for the other and have an identical design or operation of both technologies, this research must design and implement two systems. For this research, there is no need to have two fully functioning smart home systems, therefore two digital prototypes are going to be designed. The design of both smart home system prototypes is described in the following chapter.

3.2 DESIGN OF SMART HOME SYSTEMS

In designing both prototypes, a centralized, server-based smart home system architecture is considered to apply its design and functionality to the Ethereum-based prototype. This ensures that both prototypes use the server and the blockchain with the same functionality and that the prototypes are comparable. Furthermore, it ensures that the described security issues related to smart home systems are still applicable.

The article by Ahmed et al. (2021) describes the structure of a smart home system based on a server which illustration can be seen in Figure 2. It is described that the home automation devices or the sensors continuously send data to the server. The user can execute queries against the server to get the data from the sensors and control the smart home system. The user is also able to send commands directly to the home automation devices or sensors. The communication between several sensors and their master microcontrollers is wired, but the communication between the remaining components is wireless.

The described flow from the article by Ahmed et al. (2021) is shown as a sequence diagram in Figure 4. To simplify the flow, it is assumed that the sensors that send data to the server are different from the home automation devices that ultimately perform tasks such as turning the heating off or on. These devices are called controllers. Furthermore, it is assumed that the controller does not return anything specific like text

saying that the heating is on or off because for example, when the heating is turned on, the user should realize this physically. Additionally, for simplification, it is assumed that there is only one sensor, one controller, and one user, with wireless communication between all participants.

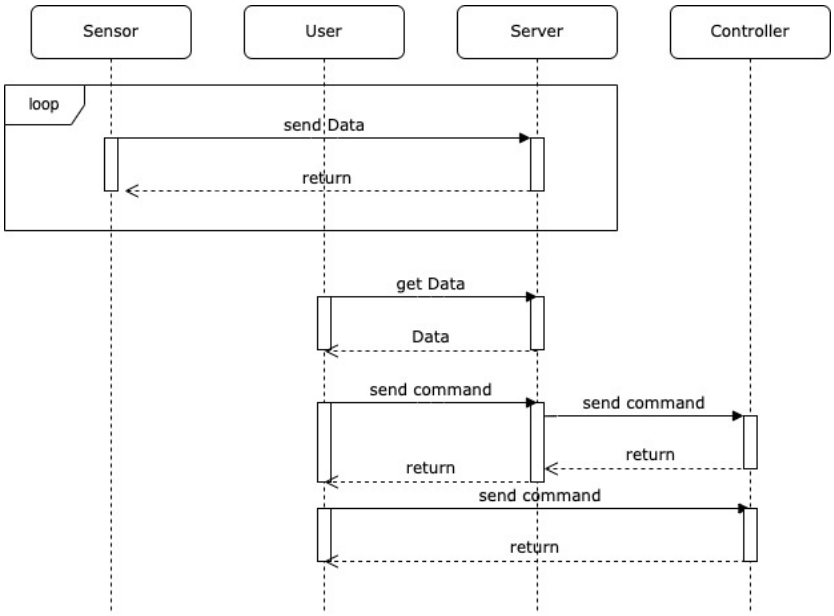


Figure 4 Sequence Diagram of Centralized, Server-based Smart Home System according to the Article of Ahmed et al. (2021)

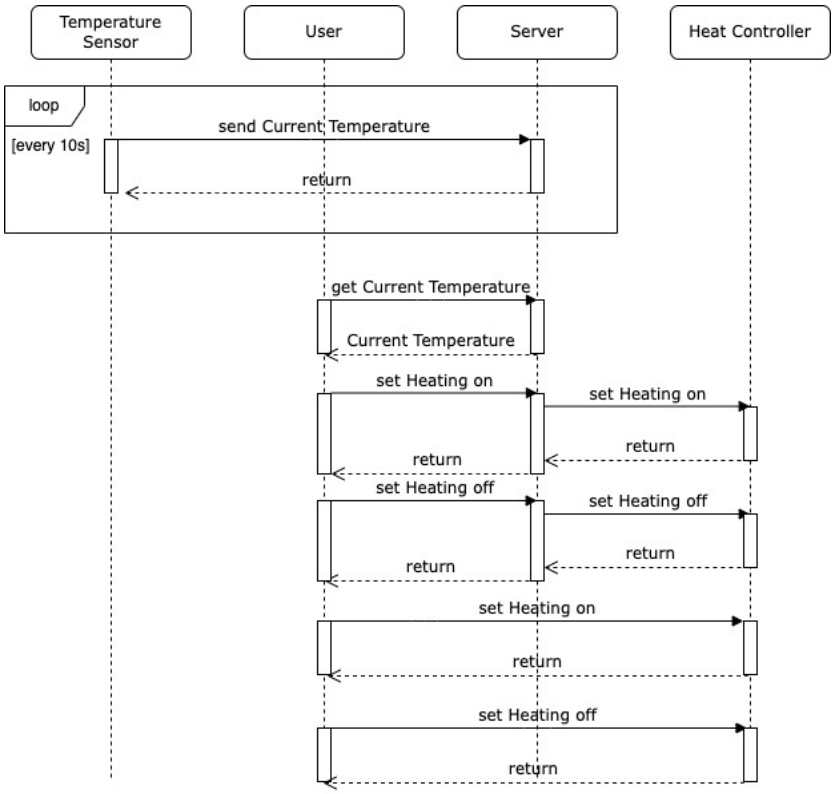


Figure 5 Sequence Diagram of first Design of proposed Centralized, Server-based Smart Home System

To concretize the proposed centralized, server-based smart home system by Ahmed et al. (2021), it is assumed that the sensor is a temperature sensor, and the controller is a heat controller which can turn the heating off and on. Furthermore, it is assumed that the temperature sensor sends a temperature to the server every 10s and the server saves this temperature. But as only the current temperature is of interest to the user, the server does overwrite the current temperature with the new value from the temperature sensor every time it is received. Additionally, the user can either ask the server to set the heating on or off or directly ask the heat controller. To test the pure technology during the experiment, additional security mechanisms like the implementation of authentication are not used. The concretized sequence diagram of the centralized, server-based smart home system can be seen in Figure 5.

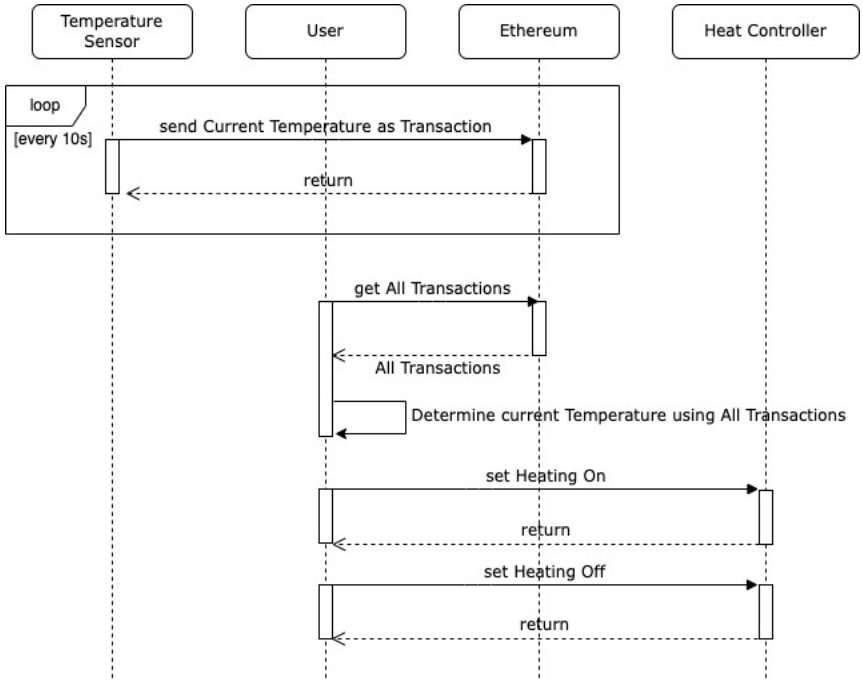


Figure 6 Sequence Diagram of a simple Ethereum-based Smart Home System

For the development of the design of the second prototype, it is important to consider that the server and Ethereum are fundamentally different. The server can receive requests from a user and answer or forward them (Ahmed et al., 2021), whereas a blockchain and thus Ethereum is described as a distributed database that stores transactions (Ozyilmaz & Yurdakul, 2019, p. 2). For the design of the second prototype, this means that Ethereum can only be seen as a storage medium. This means that Ethereum can only store temperature from the temperature sensor. When designing the centralized, server-based approach, it has been assumed that it only keeps the current temperature and not the total amount of temperatures sent. Therefore,

there is a difference in the design of both prototypes here. The user accesses the Ethereum blockchain and sees the set of all transactions and independently picks out the last and therefore most recent transaction and takes the temperature from it. Additionally, the user cannot request the Ethereum blockchain to forward their request to turn the heater on or off to the Heat Controller. This means that the user must address the heat controller directly. This results in the sequence diagram in Figure 6.

To make the two prototypes more comparable, the centralized, server-based smart home system prototype must be adapted so that it only allows the heating to be switched on or off by directly addressing the heat controller. Additionally, the server must store all temperatures received from the temperature sensor likewise to the Ethereum-based Prototype. Therefore, the user must determine the current temperature themself. Accordingly, the final design of the centralized server-based prototype is shown in Figure 7.

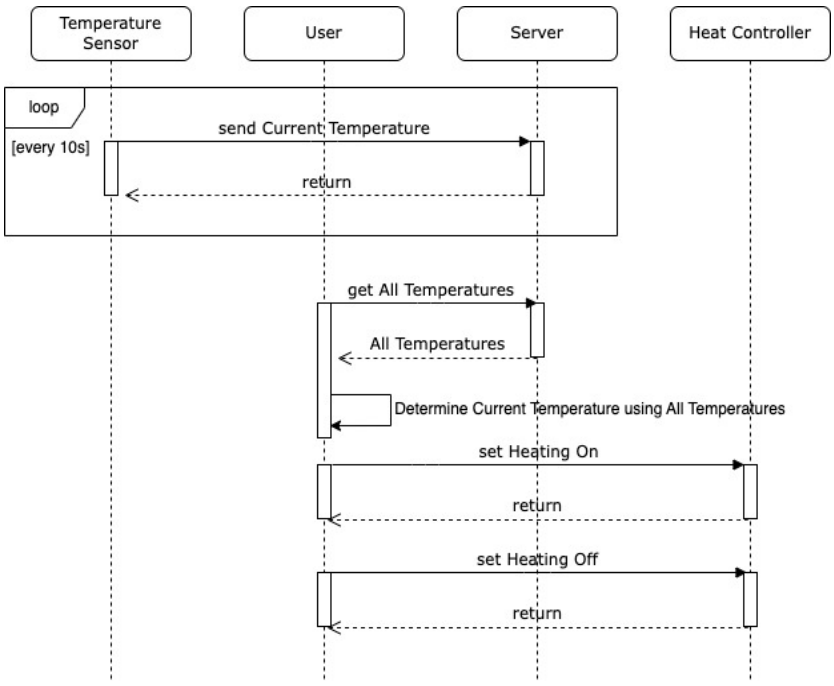


Figure 7 Sequence Diagram of the final proposed Centralized, Server-based Smart Home System

While the central server-based smart home system does not have any built-in security mechanisms, the Ethereum blockchain has the principle of digital signatures (Aung & Tantidham, 2017). This means that each participant in the blockchain must sign their transactions with their secret private key. This can be used to prove that the user who claims to be the sender of a message is the actual sender of the message. This means

that each participant in the blockchain, i.e., the temperature sensor and the user, needs a public and private key for encryption to be able to send transactions to the blockchain. Since only the sensor sends messages or transactions to the blockchain, only the sensor must sign its messages. Since the server-based prototype does not use such a mechanism and the similarity of the prototypes must be maintained, signed messages are also only used for sending transactions to the blockchain. If the user addresses the heat controller, then it will continue to operate without signed messages. Moreover, the user will use the digital signature mechanism as it is part of the Ethereum technology and will only consider the transactions where the temperature sensor is the sender of the temperature transaction according to its digital signature. This difference to the central server prototype is consciously perceived to use the core technology Ethereum correctly.

According to Aung & Tantidham (2017), there is a crucial difference between private and public blockchains. In a public blockchain, anyone can join the network and participate in the mining process, whereas, in a private blockchain, there is only one specific owner who can trade the blockchain network. Therefore, with a private blockchain, no mining mechanism is needed. This means that depending on the decision about whether the smart home system with Ethereum blockchain uses a private or public blockchain, the design must be adjusted accordingly. The smart home system should be accessible from the Internet to control it from outside and inside the home, and only the residents and users of the system should be able to access it. Since the literature research did not reveal any information showing that a private blockchain is accessible from the Internet or the other way around, and it would not matter if someone mines blocks other than the only current user of the network, it is assumed that the public blockchain is adequate for the smart home system. As a result, a miner must be included in the design of the Ethereum-based smart home system.

As mentioned above and described in the article by Benlamri et al. (2020), a blockchain requires a miner. Since there is only one user in the current prototype, this user will also be the miner of the Ethereum blockchain. It is also described in Benlamri et al. (2020) that the miner must find a valid nonce. Once the miner finds it, the miner is allowed to add the new block to the blockchain. Since Ethereum is used as a blockchain, Proof of Work must be implemented as the consensus algorithm (Ozyilmaz & Yurdakul, 2019, p. 3). This consensus algorithm describes solving a hard-to-compute but easy-to-evaluate problem. The article by Benlamri et al. (2020)

describes that such a problem can be the computation of a hash value with leading zeros.

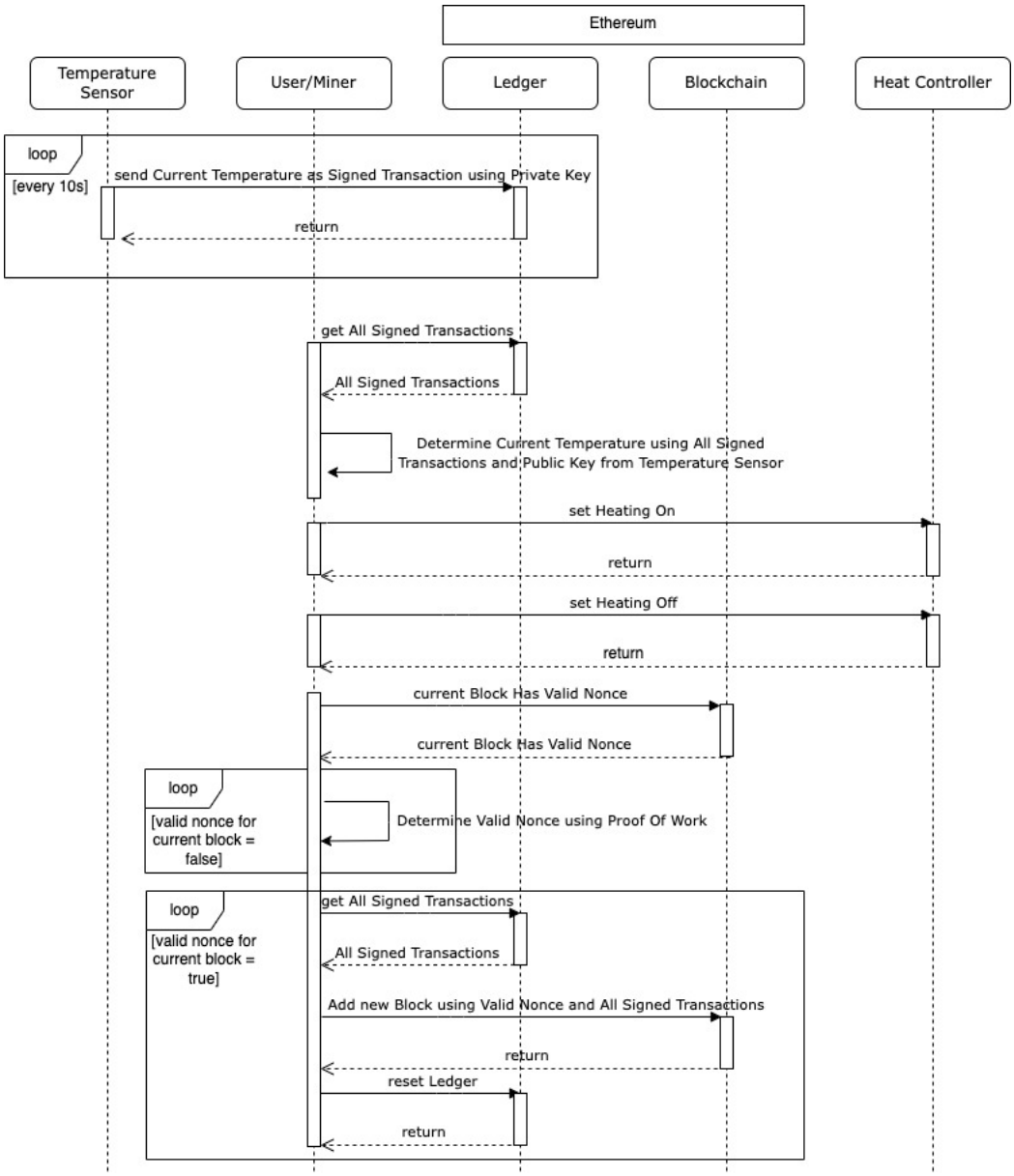


Figure 8 Sequence Diagram of Proposed Ethereum-based Smart Home System

Equally worth considering is the inclusion of a ledger in the Ethereum-based smart home system. In the article by Alex & Stephen (2018, p. 6) it is described that each user has a ledger assigned and this ledger contains all transactions that this user makes. This means that the set of all transactions that a user makes is stored in this ledger. When a miner mines a new block, these transactions are transferred to the blockchain and stored there (Benlamri et al., 2020). This means that since the temperature sensor is the only contributor to the blockchain, there must be exactly one ledger that records

the transactions, in other words, the temperatures sent to the blockchain before they are written down in a block and thus included in the blockchain.

The mentioned factors result in the final sequence diagram in Figure 8 for the Ethereum-based smart home system. The design in Figure 8 assumes that only the most recent transactions are relevant for evaluating the current temperature. This means that the user only needs to check the ledger of the temperature sensor. The history data stored in the Ethereum blockchain is not of interest when finding out the current temperature.

Since a smart home system must be accessible via the Internet by default to monitor and control the system not only at home in the home network but also while the user is away from home, it is assumed that both prototypes are accessible via the Internet.

After designing the prototypes in this chapter, the next chapter designs the experiments which are executed using the previously designed prototypes to answer the research question.

3.3 DESIGN OF EXPERIMENT

Throughout the literature research, the article by Komninos et al. (2011, p.184-185) addresses the internal and external security risks of smart home systems. Since the research question refers to cyberattacks, the design of the experiments will focus exclusively on external threats or cyberattacks. Komninos et al. (2011, p.184) distinguish between passive and active cyberattacks and names eavesdropping and traffic analysis as active cyberattacks and masquerading, replay, message modification, and denial-of-service attack as active attack possibilities of smart home systems (Komninos et al., 2011, p. 185). On security risks from cyberattacks of Ethereum or blockchain in general, few were found during the Literature Research. Only the article by Alex & Stephen (2018, p.3) describes that blockchains have a risk of distributed denial-of-service and man-in-the-middle attacks.

This means that the selected cyberattacks should have a mix of designated passive and active cyberattacks affecting smart home systems and blockchain, so that a high variety of attack types can be covered by the experiment. To simplify the research, the number of attacks is limited to five cyberattacks.

Since only the article by Komninos et al. (2011, p. 184) deals with passive attacks and names eavesdropping as the most common security problem in open networks, it is

selected for the experiments (Komninos et al., 2011, p. 185). In addition, Traffic Analysis is also chosen because it was named as the only other passive attack, and it would not be representative of the variety of cyberattack types to choose only one passive attack out of five attack types. Since the article by Komninos et al. (2011, p. 185) and the article by Alex & Stephen (2018, p. 3) both mention the denial-of-service attack as a security risk for smart home systems and blockchain, this active attack is selected as the third attack. Komninos et al. (2011, p.185) mentions the Message Modification and Replay Attack. In the replay attack, the attacker tries to gain access to messages from two authorized users to retransmit them later to gain access to the services of the authorized users. In the Message Modification attack, the attacker tries to delete, add, or modify message contents. Not only is this quite like a replay attack, but Komninos et al. (2011, p. 185) also describes that the message modification attack often leads to denial-of-service attacks, which has already been included as an attack type for the experiments. In addition, the article by Alex & Stephen (2018, p. 3) describes the man-in-the-middle attack, where the attacker tries to interpose himself between two legitimate parties, intercept their messages to obtain sensitive data, and adapt these messages. This attack includes message changing, which the Message Modification attack also does, as well as message interception and retransmission. Therefore, the man-in-the-middle attack goes even one step further than the Message Modification and Replay attack, as the attacker tries to completely interpose himself between the communication of two legitimate parties. Therefore, this attack is chosen as the fourth attack type for the experiments. Lastly, there remains the masquerading attack mentioned by Komninos et al. (2011, p. 185), which aims to make the attacker impersonate a legitimate user to obtain sensitive information and obtain privileges. Since this form is not covered by any other attack type, the masquerading attack is selected as the fifth and final attack type.

All five cyberattacks will be executed once against both prototypes so that a total of five cyberattacks per prototype can be defeated.

The research question addresses the question of whether an Ethereum-based smart home system can withstand more different types of cyberattacks than a centralized server-based smart home system. This means that all five selected cyberattacks must be fired against both smart home systems and it must be measured which of the two systems withstood how many attacks. As part of the experiment, all five cyberattacks

will be executed against both prototypes once, so that a total of five cyberattacks per prototype can be defeated.

There are different aspects to be considered for the execution of the experiments. First, the execution time of all attacks is limited to 100s and only the results of the 100s are considered in the evaluation of the attacks. Second, for the execution of all attacks, it is assumed that the interfaces of the systems are known. However, it is not known who is using these interfaces. This means that the respective attacks should be performed as follows: In the eavesdropping attack, the attacker should listen to all interfaces and save their message traffic over the 100s of the attack. Should this succeed without the smart home system noticing, then the data collected from the eavesdropping attack can be used as the basis for the traffic analysis attack. If the eavesdropping fails, then the traffic analysis cannot be performed and is automatically considered unsuccessful. However, if the eavesdropping attack succeeds, then the traffic analysis attack must extract sensitive information from the data, such as credentials, operating methods, or critical infrastructure information of the system, which could be used for subsequent attacks. In the first active attack, the denial-of-service attack, the attacker must send many messages to the system to compromise its functioning. Since all interfaces are known, all interfaces could be overloaded, however, I only see two interfaces as critical to the functioning of both systems. One is the interface to the controller turning the heating on and off, and the other is the interface to the server or blockchain, which receives the temperatures. Because even if the user can get the temperatures from the server or blockchain, then the information does not benefit the user if this data is incorrect because the server or blockchain receives incorrect data due to the denial-of-service attack. In the second active attack, the man-in-the-middle attack, the attacker in the experiment should try to interpose himself between two instances and get, for example, the current temperature from the sensor without the server or blockchain also receiving this message. The attacker can then forward the message but also forward it altered. The final attack of the experiment is the masquerading attack. Here, an additional user will send the same messages to the existing interfaces as the actual legitimate party would send. This means that the attacker will imitate the messages of other parties like the temperature sensor or the user and send the same messages.

As mentioned before, an attack is considered successful if its definition has been met, even if only in parts. This means that an eavesdropping attack is considered successful if the attacker can monitor only parts of the traffic between the components of the

smart home system without them noticing. The system could be aware of the monitoring if the attacker leaves traces when monitoring the traffic and thus can be proven to have acted in the system, such as log entries, because every user of the network is possibly registered. The Traffic Analysis attack also deals with spying on the communication of the smart home system, but the attacker tries to analyse the communication and extract sensitive information such as credentials from captured messages (Komninos et al., 2011, p. 184). However, even if this information is encrypted, the attacker can infer the structure of the system or critical infrastructure through communication between the parties, which can then be abused for active attacks. This means that traffic analysis is considered successful if either sensitive information, crypted or encrypted, can be uniquely identified from the communication or if the communication analysis can be used to conclude how the system works. As mentioned above, it is sufficient if the traffic analysis can only analyse parts of the system. In a denial-of-service attack, the goal is to make the network unreachable or to limit its availability (Komninos et al., 2011, p. 185). The attacker sends countless messages to the system to overload it. This means that this attack is considered successful when the user is no longer able to use parts of the system's functions, such as getting the current temperature from the server or blockchain or turning the heating off or on. With the man-in-the-middle attack, the attacker tries to intercept the communication between two parties and thus be able to watch the messages but also to forward them in a modified way (Alex & Stephen, 2018, p. 3). This attack is considered successful if, for example, the attacker can interpose himself between the sensor and the server and receive the message from the sensor to the server and forward it original or modified. Finally, there is the masquerading attack, where an attacker tries to be a legitimate user and obtains unauthorized privileges and thus sensitive information or access to services (Komninos et al., 2011, p. 185). This attack is considered successful if another user in the system has the same or even partially the same privileges as the existing user of the system and thus can, for example, command the heat controller to turn the heating off or on.

After all, cyberattacks have been executed against both systems, the number of unsuccessful cyberattacks is compared. If the Ethereum-based Smart home system prototype has more defended cyberattacks, then it means for the research question that the Ethereum-based prototype can defeat more different types of cyberattacks. If the centralized server-based smart home system prototype has defended the same number

or more cyberattacks, then for the research question it means that the Ethereum-based smart home system prototype cannot defeat more different types of cyberattacks than the centralized server-based smart home system.

In summary, the experiments for the research were designed in this subchapter. Various aspects such as the existing literature or the research question were incorporated into the design of the experiments. It was determined which cyberattacks and in which way they should be carried out. Boundaries, assumptions, and settings of the experiments were also determined, which are crucial for the execution of the experiments. It was also defined when a cyberattack is considered successful or unsuccessful and what meaning the results of the experiments have for the Research Question. This process is illustrated in the following flow chart for easier understanding.

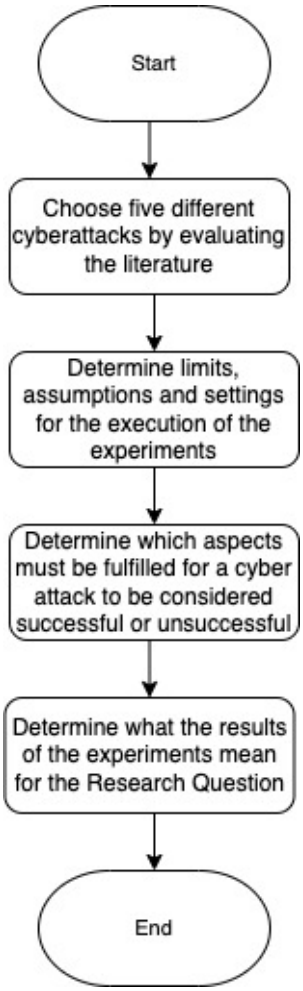


Figure 9 Flow Chart describing the process of Designing the Experiments

4 IMPLEMENTATION AND EXPERIMENT

After setting the foundation for this research in the previous chapter by defining the methodology, designing the different smart home systems and the related experiments used for data collection, the theory is implemented in this chapter. Therefore, this chapter describes how the centralized, server-based and Ethereum-based smart home prototype is built. Additionally, the execution of passive and active cyberattacks against the respective prototypes and their outcomes are described.

4.1 IMPLEMENTATION OF PROTOTYPES

This subchapter deals with the implementation of the prototypes that will be used to run the experiments. Therefore, two different smart home system prototypes are created: a centralized server-based smart home system prototype and an Ethereum-based smart home system prototype. Their components, design, and functionality are described in the following sections.

4.1.1 CENTRALIZED, SERVER-BASED SMART HOME SYSTEM PROTOTYPE

This subsection deals with the description of the implementation of the design for the centralized, server-based smart home system. This system consists of a temperature sensor, a heating controller, a user, and a central server. The design and operation of the sensor, controller, user, and server are explained in the following sections. It is then explained how these components interact to form a fully functional smart home system.

4.1.1.1 TEMPERATURE SENSOR

The temperature sensor is responsible for measuring the current temperature and sending it to the server. As the sensor is part of a prototype, it does not measure the actual temperature but, as shown in Figure 10, it generates random temperature values in the range of 10 to 30.

Therefore, a timer is started every 10000ms or 10s, which triggers a module that produces random numbers. As this module produces decimal numbers, a round module

is used to round the number to an integer. Afterwards, the number is put into an object, which only shows the temperature. The object is formatted to json and as soon as the generated temperature changes, a publish module is triggered by the on-change module, which sends the generated json object to the channel "server-temperature". As all temperature sensor related modules are packed into one temperature sensor module, which is used in the final prototype, an output module is added, so that the current temperature can be checked from the outside of this module. Furthermore, the sensor communicates wirelessly with the server via publish and subscribe function in the ETH.build application, which simulates the functionality of HTTP.

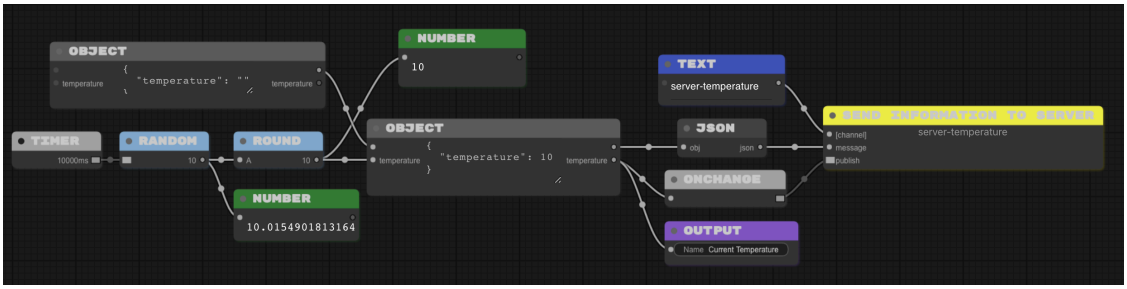


Figure 10 Structure of Temperature Sensor for Centralized, Server-based Prototype (Screenshot)

4.1.1.2 HEAT CONTROLLER

The responsibility of the heat controller is to control the heating system. Since the controller is part of a prototype and does not turn a heater off or on, this functionality is simulated by having the output of the controller display a text describing whether the heater is currently on or off. The implementation of the heat controller can be seen in Figure 11.

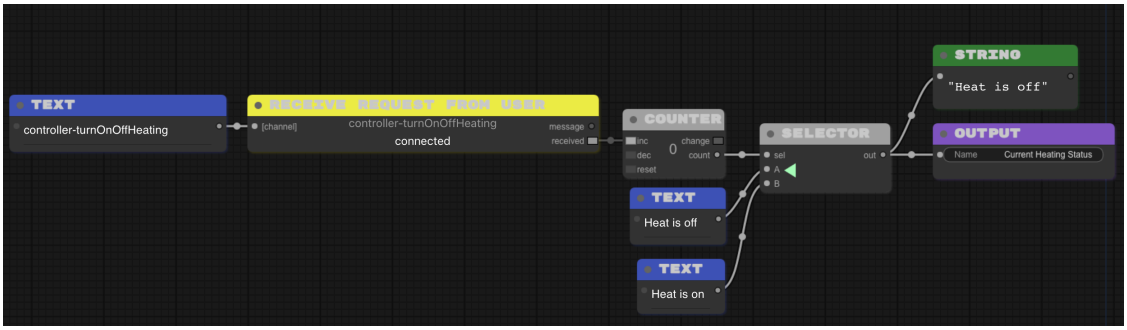


Figure 11 Structure of Heat Controller for Centralized, Server-based Prototype (Screenshot)

The heat controller operates by using a subscribe module which is set to the channel "controller-turnOnOffHeating". As soon as a message is published to this channel, a

counter is incremented. This counter is accessed by a selector module, which selects either the text "Heat is off" or "Heat is on" depending on the counter. As soon as the selector receives an even number from the counter, it selects "Heat is off". When the counter changes and shows an odd number, the selector chooses "Heat is on". This ensures that the heating is switched off and on each time the controller is triggered. Additionally, all classes related to the heat controller are packed into one heat controller module. Therefore, an output function is added, so that the current heating status can be seen from outside the module when using it in the prototype.
The controller as well as the sensor works wireless and acts via the publish and subscribe function of the ETH.build application.

4.1.1.3 USER

The user also named Alice can perform two different actions in the smart home system. First, it can request all temperatures from the server and use it to determine the current temperature and second, ask the controller to turn the heating on or off.
Figure 12 shows the implementation for requesting the server for all temperatures. Consequently, the implementation has an input module that can be used to trigger this function from outside. Since all the user's functions have been combined into one user module and therefore can be controlled from the outside by using the input function. If a signal is sent to this input, the on-change module reacts and publishes a message to the channel "server-currentTemperatures". The message contains random text which is required by the ETH.build application to work and send the request properly.

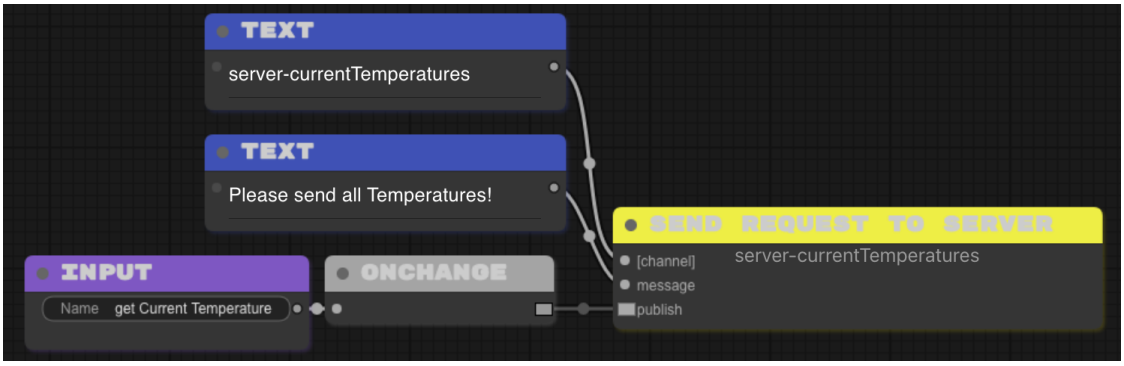


Figure 12 Structure of User requesting the current temperature from the Centralized, Server-based Prototype (Screenshot)

When the server receives the request regarding the current temperatures, it will send a response to the channel "user-currentTemperatures" containing all saved temperatures. Therefore, the user module contains a subscribe to this channel and as soon as a

response arrives to this channel, parses the response json into an object. Then the latest temperature out of the received list of temperatures is displayed using an output module. The use of that output module makes it possible to access the current temperature from outside the user module. The functionality of receiving the current temperature from the server can be seen in Figure 13.

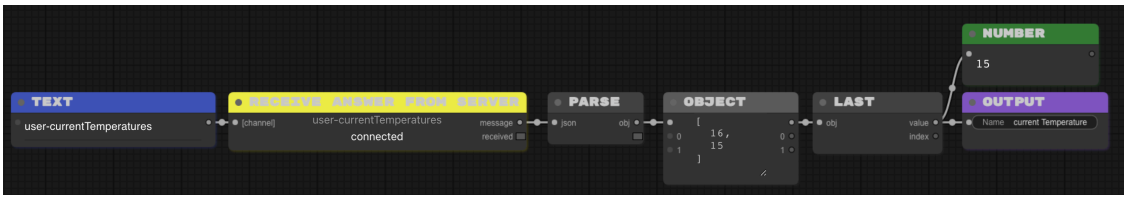


Figure 13 Structure of User receiving all temperatures from the Centralized, Server-based Prototype (Screenshot)

The second function that the user can perform is to change the heat status from off to on and vice versa. The implementation for this can be seen in Figure 14. For this purpose, the user sends a request to the channel "controller-turnOnOffHeating", which is subscribed by the heat controller. As soon as the heat controller receives the request, it turns the heating off or on.

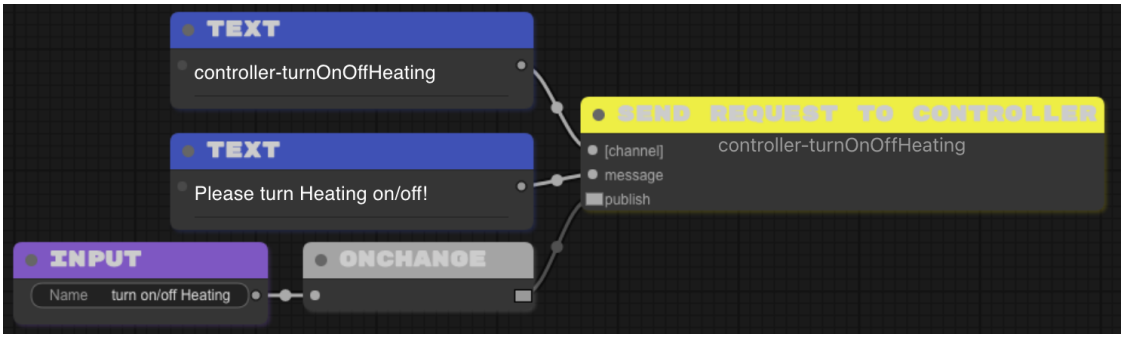


Figure 14 Structure of User requesting a change of the current heat status from the Centralized, Server-based Prototype (Screenshot)

4.1.1.4 SERVER

The server can handle two different scenarios. First, it receives the current temperature from the sensor and saves it and second, it can respond to requests that ask for the list of current temperatures.

Figure 15 shows the implementation for retrieving the current temperatures. The server subscribes to the channel "server-temperature" to where the temperature sensor sends the generated temperature values and processes the received json into an object. Then, the temperature is passed to a list so that all temperatures received from the

temperature sensor are saved. The newest received temperature is forwarded to the output function so that the temperature can be seen from outside of the server module. Additionally, the list of temperatures is stored in a variable so that it can be used by other functions as well.

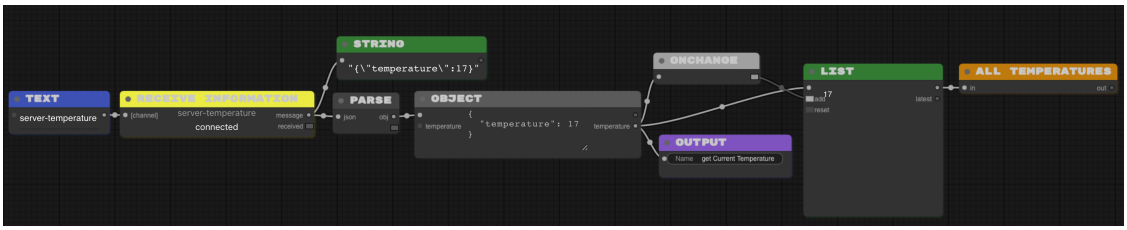


Figure 15 Structure of Central Server receiving the current Temperature from Sensor (Screenshot)

Afterwards, the server receives a request for getting the list of temperatures, which is shown in Figure 16. Therefore, the server has a subscription to the channel "server-currentTemperatures". As soon as a request is sent to that channel, the server will immediately trigger the sending of the list of temperatures to the “user-currentTemperatures” channel. The list of temperatures is stored in the variable “All Temperatures” and must be parsed into json before sending as a message value. Otherwise, the receiver channel will not be able to reconstruct a list of values out of the message value.

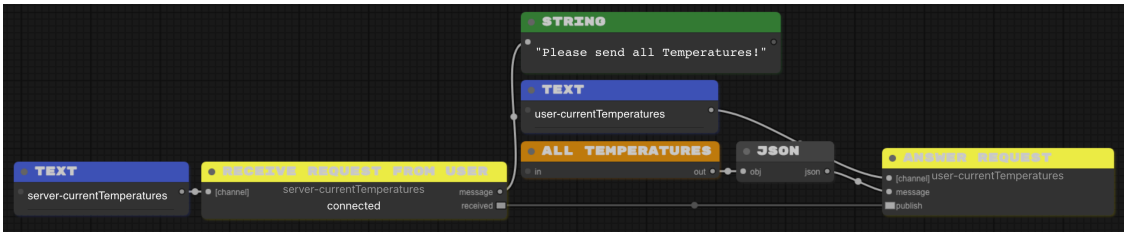


Figure 16 Structure of Central Server receiving a request of getting all temperatures (Screenshot)

4.1.1.5 STRUCTURE OF CENTRALIZED, SERVER-BASED SMART HOME SYSTEM PROTOTYPE

Following the description of the functionality and implementation of the individual components in the previous chapters, this chapter explains how these components collaborate with each other. For this purpose, Figure 17 shows the centralized, server-based smart home system prototype. It consists of a temperature sensor, a heat controller, a server, and a user module. By using the input and output functions, the individual modules can be controlled from outside. The temperature sensor currently

shows a temperature of 10 degrees Celsius, which is sent to the server. The server saves all received temperatures in a list and as soon as the button for the “get Current Temperature” function on the user is pressed, the server sends the list of temperatures to the user who evaluates the current temperature from it. Then, the current Temperature output of the user shows the correct temperature. Furthermore, at the beginning the heating is always switched off, which can be seen at the output of the heat controller. As soon as the input button on the user for switching the heater on or off is pressed, the status on the heat controller changes as the request is sent from the user to the heat controller.

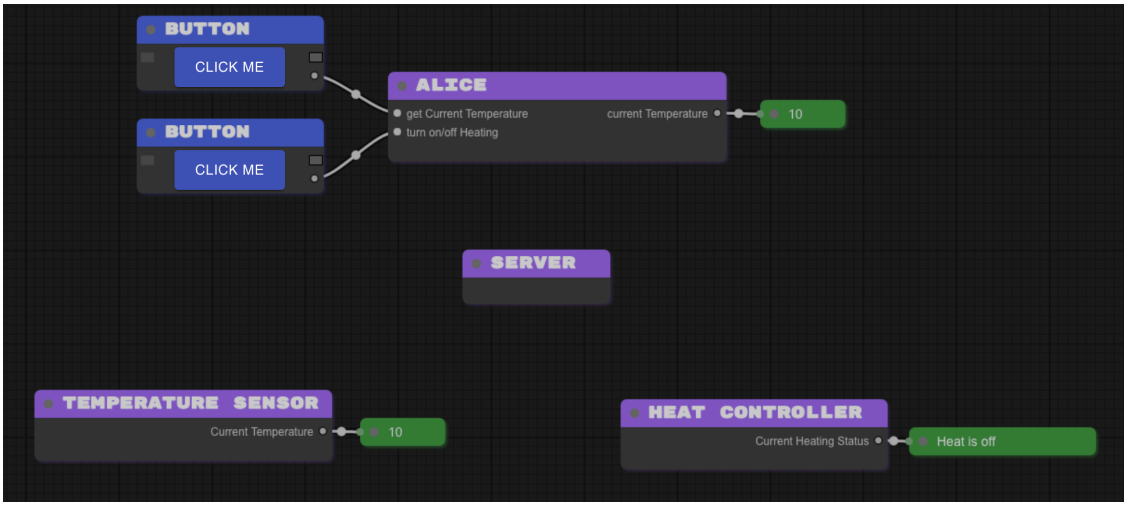


Figure 17 Structure of Centralized, Server-based Smart Home System Prototype (Screenshot)

An overview of all used publish and subscribe channels can be seen in the following Table 4 including which component sends or subscribes to it.

Table 4 Overview of all publish and subscribe channels and their sender and receiver components of Centralized, Server-based Smart Home System Prototype.

Channel	Sender	Receiver
server-temperature	Temperature Sensor	Server
controller-turnOnOffHeating	User	Heat Controller
server-currentTemperatures	User	Server
user-currentTemperatures	Server	User

4.1.2 ETHEREUM-BASED SMART HOME SYSTEM PROTOTYPE

Since the previous subsection described the functionality of the centralized, server-based smart home system prototype, this subchapter describes the implementation of the Ethereum-based smart home system prototype. The system consists of a temperature sensor, a heat controller, a user, and an Ethereum blockchain. The structure and functionality of the sensor, controller, user, and blockchain are explained in the following chapters. Subsequently, it is explained how these components interact to form a fully functional smart home system prototype.

4.1.2.1 TEMPERATURE SENSOR

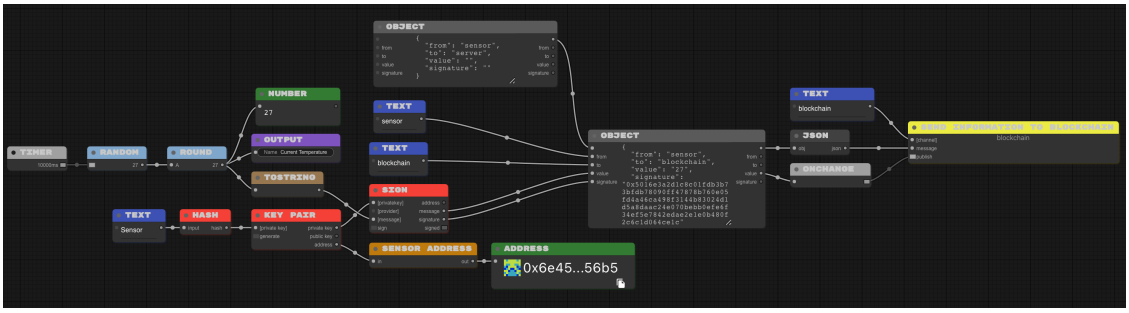


Figure 18 Structure of Temperature Sensor of Ethereum-based Smart Home System Prototype (Screenshot)

The temperature sensor of the Ethereum-based smart home prototype can be seen in Figure 18. A random module is triggered every 10000ms or 10s, which produces random numbers in the range of 10 to 30. Because this module produces decimal numbers, a round function is connected to it, which maps the randomly generated number to an integer. Since all temperature sensor-related functions are combined in one module which is then used in the smart home system prototype, the generated temperature is passed on to an output function. This allows the temperature to be gathered from the outside of this temperature sensor module. As explained in Chapter 3.2, the Ethereum-based smart home system requires each participant in the network, who is sending transactions to the blockchain, to have a public and private key pair for signing and recovering messages. Therefore, when the temperature sensor sends the generated temperature to the blockchain, the message must be signed. Consequently, a key pair is created for the sensor. When creating the key pair, a random input value is used, which is converted to a hash value and thus also forms the private key of the

sensor. By using the generated private key, the temperature message can be signed. This allows anyone who receives this message and the corresponding signature to track who signed the message and determine whether the temperature message originated from the sensor. To detect this, the address of the generated key pair is stored in a variable so that it can be used by other participants in the network. This is because the address and the public key are publicly available and known. After the message is signed, a json object consisting of a “from”, “to”, “value” and “signature” value is filled. The “from” is set by default to “sensor” and the “to” to “blockchain” for every message. These values are not meaningful in themselves, but they make it easier to understand how the prototype works and the ledger requires a “from”, “to” and “value” field in the json. As soon as the temperature changes, an event is triggered which sends the generated json to the channel "blockchain".

4.1.2.2 HEAT CONTROLLER

The heat controller in the Ethereum-based smart home system is identical to the heat controller used in the centralized, server-based smart home system prototype. The responsibility of the heat controller is to control the heating system. Since the controller is part of a prototype and does not actually turn a heater off or on, this functionality is simulated by having the output of the controller display a text describing whether the heater is currently on or off. The implementation of the heat controller can be seen in Figure 19.

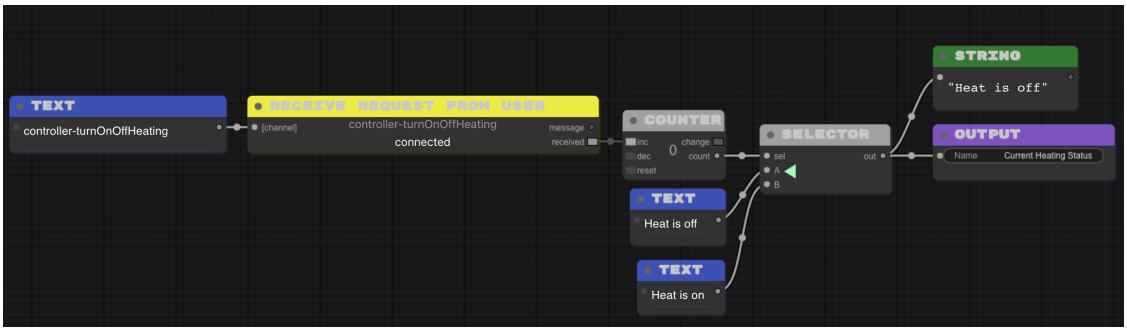


Figure 19 Structure of Heat Controller of Ethereum-based Smart Home System Prototype (Screenshot)

The heat controller operates by using a subscribe module that is set to the channel "controller-turnOnOffHeating". As soon as a message is published to this channel, a counter is incremented. This counter is accessed by a selector module, which selects either the text "Heat is off" or "Heat is on" depending on the counter. As soon as the

selector gets an even number from the counter, it selects "Heat is off". When the counter changes and shows an odd number, the selector chooses "Heat is on". This ensures that the heating is switched off or on each time the controller is triggered. Additionally, all classes related to the Heat Controller are packed into one Heat Controller module. Therefore, an output function was added, so that the current heating status can be seen from outside the module when using it in the prototype.
The controller as well as the sensor works wireless and acts via the publish and subscribe function of the ETH.build application.

4.1.2.3 USER

The user, also called Alice, can send requests to the heat controller to switch the heating on and off and can read all temperatures from the blockchain and determine the current temperature. The implementation for sending requests to the heat controller is shown in Figure 20 and in Figure 21 the implementation for examining the current temperature.
Sending requests to the heat controller works by sending a message to the channel "controller-turnOnOffHeating". Since the user is an independent module, which can be operated from outside, the sending of this message is triggered by an input function. As soon as this changes, a message is sent to the controller.

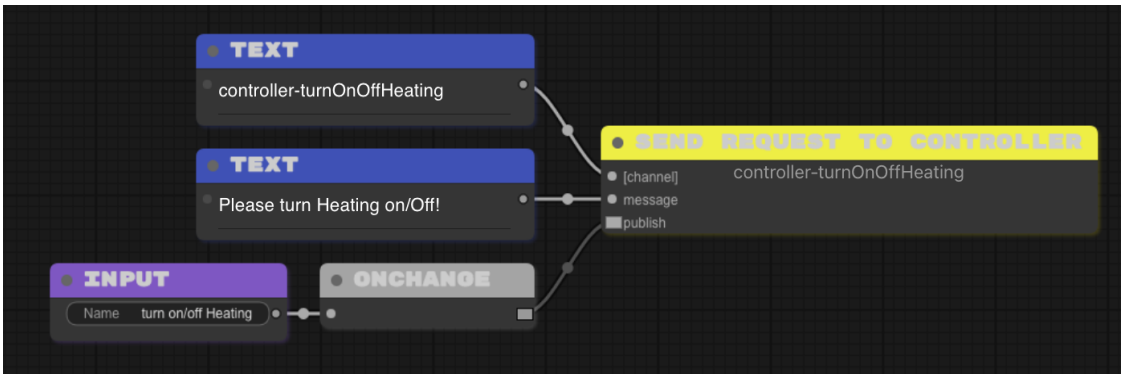


Figure 20 Structure of User Sending a Request to the Heat Controller for changing the heat status in Etherum-based Smart Home System (Screenshot)

For examining the current temperature, Alice accesses the list of current transactions stored in the variable "Current Transactions". She has access to these transactions as she has access to the blockchain. The list of transactions is filtered so that only transactions containing the string "sensor" are kept in a list. Since the transactions of the sensor contain a from = "sensor", all transactions will pass the filtering. However,

if other sensors or parties transmit data to the blockchain, then this filtering is significant. Afterwards, the last or most recent object is taken from the list. The value and signature are used with a recover function to get the address of the creator. This address is compared with the public address of the sensor to ensure that the transaction originates from the temperature sensor. If the result of the comparison is true, the value of the object, i.e., the temperature, is displayed as the user's output.

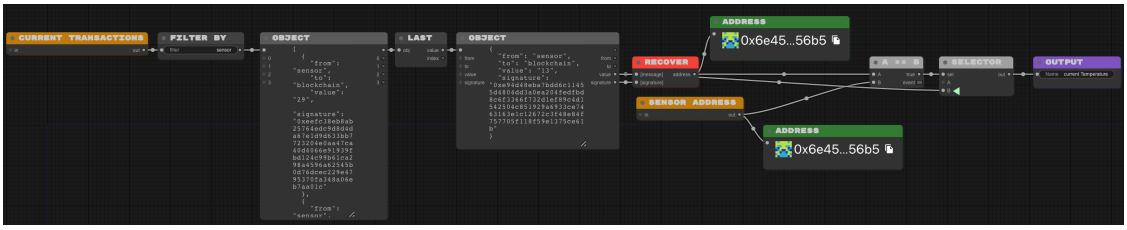


Figure 21 Structure of User determining the current temperature in Ethereum-based Smart Home System (Screenshot)

4.1.2.4 ETHEREUM BLOCKCHAIN

The Ethereum blockchain consists of a ledger, a miner, and a limited number of blocks, which was limited to two when the blockchain was initiated since the ETH.build application cannot automatically generate new blocks.

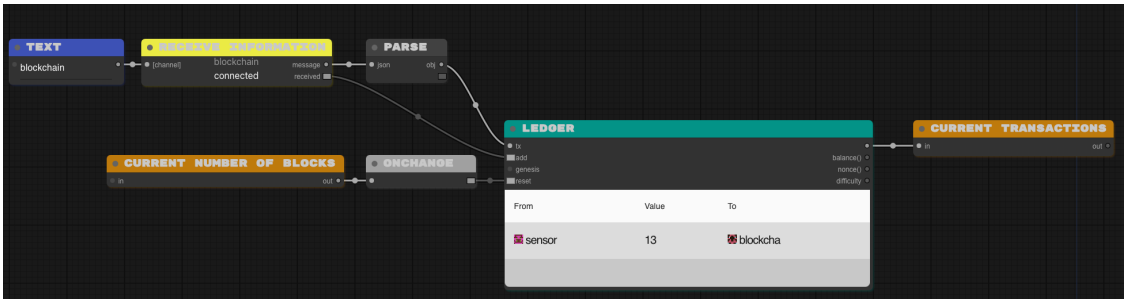


Figure 22 Structure of Ledger receiving the current temperature in Etherum-based Smart Home System (Screenshot)

The ledger, which can be seen in Figure 22, is populated with the information from the sensor by accessing the channel "blockchain". The ledger has a subscription on the mentioned channel and receives the transactions there, which consist of “from”, “to”, “value”, and a “signature” value. The received json object is parsed and added to the ledger. The quantities of all contained transactions of the ledger are stored in the variable "Current Transactions". This variable is globally accessible, as participants in the prototype have access to the blockchain and can therefore see all transactions. The transactions located in the ledger are the transactions of the block, which is currently

mined. Therefore, the ledger is reset as soon as the "Current Number of Blocks" changes because a block has been mined and a new one is started. Once a block has been mined, the past transactions are fixed and can no longer be adjusted.

Since Alice is the only user in the smart home system, she is also the only miner in the blockchain. Therefore, Alice needs a public-private key pair, which is generated by hashing a random input text and inserting this as a private key to a key pair generator module. The public address of Alice is entered into a variable as all participants in the network should know the address and therefore can access it via this variable. The implementation can be seen in Figure 23. The usage of Alices' address is described in the following paragraph.

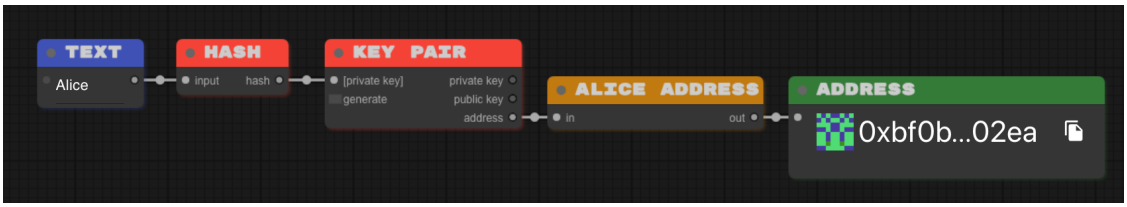


Figure 23 Structure of Public-Private Key Pair Creation for Alice in Ethereum-based Smart Home System (Screenshot)

As Alice is the miner of the Ethereum blockchain, a miner module which can be seen in Figure 24 must be used. Alice's address from the global variable is entered as the Address input into the miner function. Another input is the “Current Block” value which is a variable set by the “Set Current Block” module and always corresponds to the current block of the blockchain. The third input is a variable called “Current Transactions”. This variable is filled with all transactions from the ledger. The output of the miner is a valid nonce that can be used to mine a block.

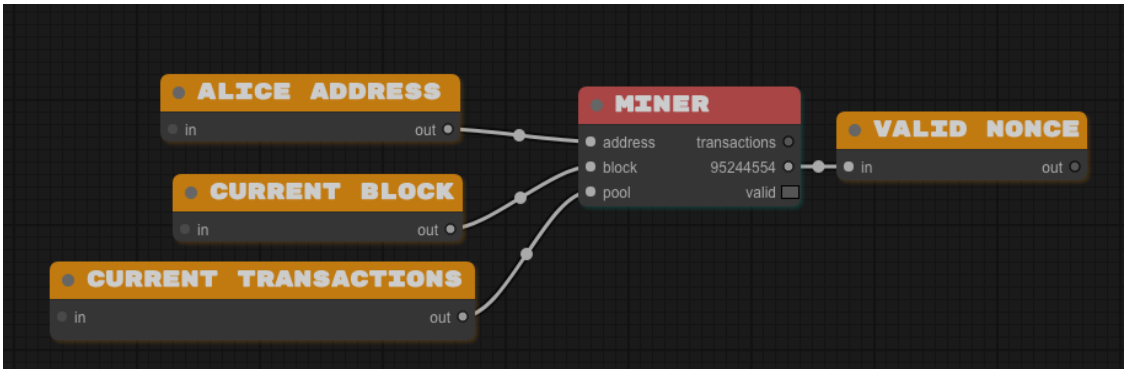


Figure 24 Structure of Mining in Ethereum-based Smart Home System (Screenshot)

As the ETH.build application does not support the generation of new blocks, the maximum number of blocks was limited to two as can be seen in Figure 25. However,

the number of blocks does not limit the functionality of the Ethereum blockchain. Each block apart from the first block needs a parent block which is the previous block. Each block is filled with the current transactions from the ledger depending on if the block is currently mined. Therefore, a “Get Transactions for Block” module was created which determines if the current block depending on the given block number is mined. The first block starts with the number zero, and the second block has the number one. For example, the “Get Transactions for Block” module will only return the list of current transactions to the block, if the number of the block matches the currently mined block. The same mechanism is used for the module “Get Nonce for Block” so that only the currently mined block receives a valid nonce as soon as one was found by the miner. Also, the module “Set Current Block” was created so that this module contains the current block number and block and as soon as a block is mined, it forwards the mined block as a parent block to the next block.

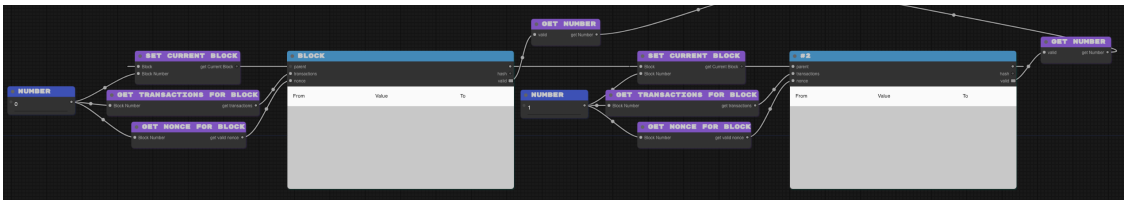


Figure 25 Structure of Block Connections in Ethereum-based Smart Home System (Screenshot)

Finally, a “Get Number” module has been introduced which implementation can be seen in Figure 26. All blocks create an event as soon as the block becomes valid and forwards it into this module. The output of the module is a number. Both output numbers from the two modules will be added and according to the resulting number changing a counter is increased. The counter number is the current number of blocks. So, when the first block is mined, then the number from the module changes which also changes the sum of both “Get Number” modules which increases the counter. The counter is changing from zero to one showing that one block was created.

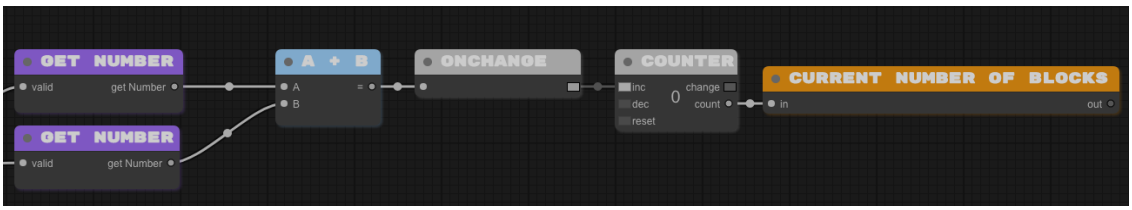


Figure 26 Structure of Logic Examining whether a block is mined n Ethereum-based Smart Home System (Screenshot)

In Figure 27, the implementation of the previously described “Set Current Block” module is shown. It uses the given block number and current number of blocks and compares them. In the beginning, the current number of blocks variable is set to zero and when this module is used at the first block, the input block number is zero as well. Therefore, both number match and result in a Boolean. This Boolean is used as a selector. When the value is false, then the selector selects the A input otherwise B. The A input is empty, but the B input receives an input block which is the current mined block. In the described example, it is the first block. This block is forwarded as an output of this module and set to a variable called “Current Block”. This module is needed as the prototype must ensure that as soon as a block is mined, the next block will receive the mined block as a parent.

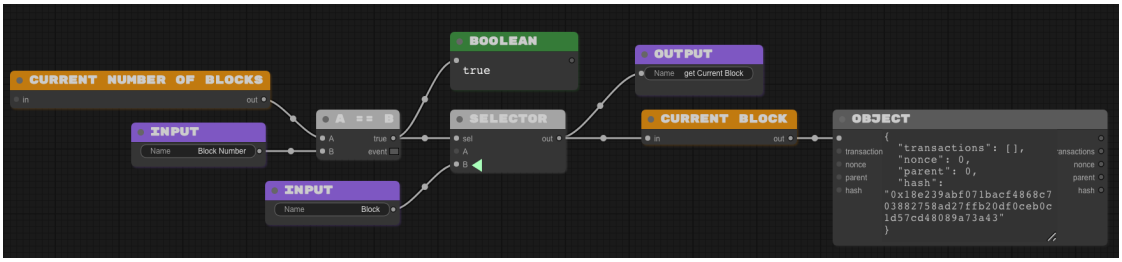


Figure 27 Structure of "Set Current Block" Module n Ethereum-based Smart Home System (Screenshot)

The module “Get Transaction for Block” is shown in Figure 28 and feeds the block with the current transactions from the ledger. Therefore, it uses the entered block number and the current number of blocks variable again for validating if the block is the currently mined block. If both matches, it will return true and make the following selector selecting the current transactions variable, which contains all transactions from the ledger.

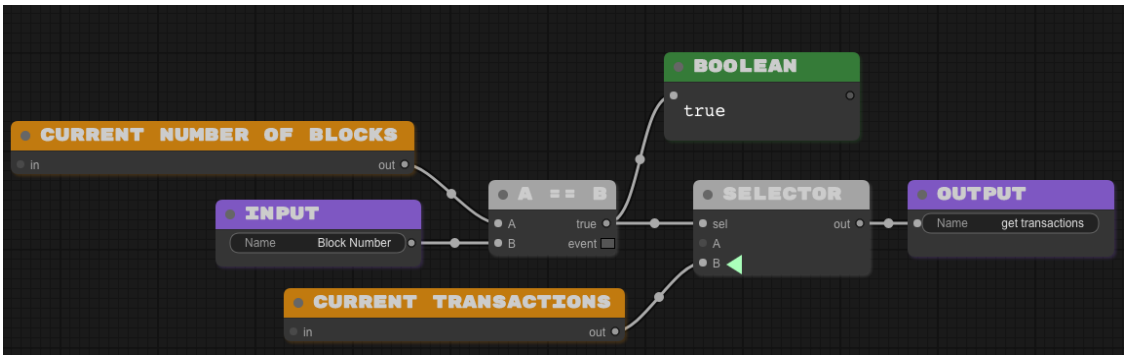


Figure 28 Structure of "Get Transaction for Block" Module n Ethereum-based Smart Home System (Screenshot)

The “Get Nonce for Block” module is shown in Figure 29 and works identically to the two previous modules. It uses the given block number and the current number of blocks and compares them. When both numbers are equal, then a selector will select the value from the valid nonce variable coming from the miner module. Here the A input from the selector module must be another number due to the functioning of the selector. When the block numbers are not matching and the selector is set to A input, then the possibly previously selected B input value will remain if the A input does not have a value. This is not problematic with the other modules but without this workaround, this module would not work. Therefore, the module returns zero by default and the valid nonce if the block is to be mined.

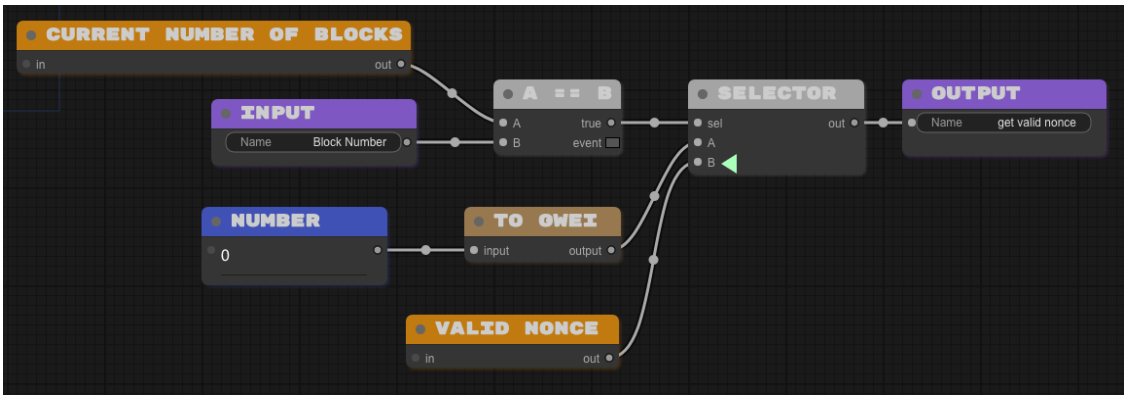


Figure 29 Structure of "Get Nonce for Block" Module n Ethereum-based Smart Home System (Screenshot)

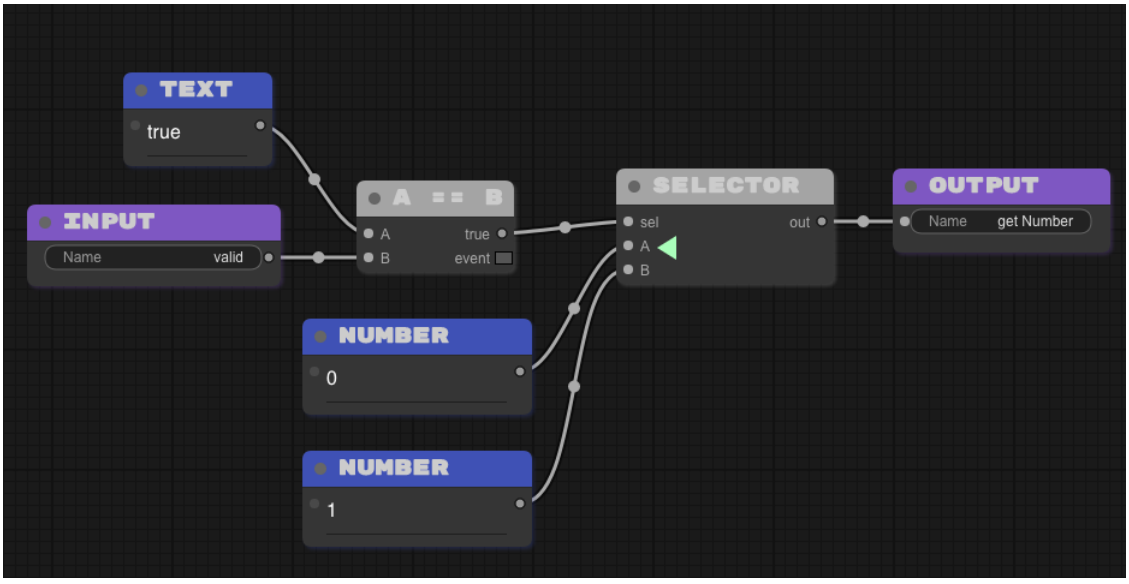


Figure 30 Structure of "Get Number" Module in Ethereum-based Smart Home System (Screenshot)

The last module is called “Get number” and its implementation is shown in Figure 30. Its purpose is to change from zero to one as soon as the input valid has received an event from a block. This is part of the logic to automatically increase the current number of block variable. The chosen numbers are not meaningful. It only must be ensured that the number changes.

4.1.2.5 STRUCTURE OF ETHEREUM-BASED SMART HOME SYSTEM PROTOTYPE

Since the previous subsections described the individual components of the Ethereum-based smart home system prototype, this chapter explains how these components interact.

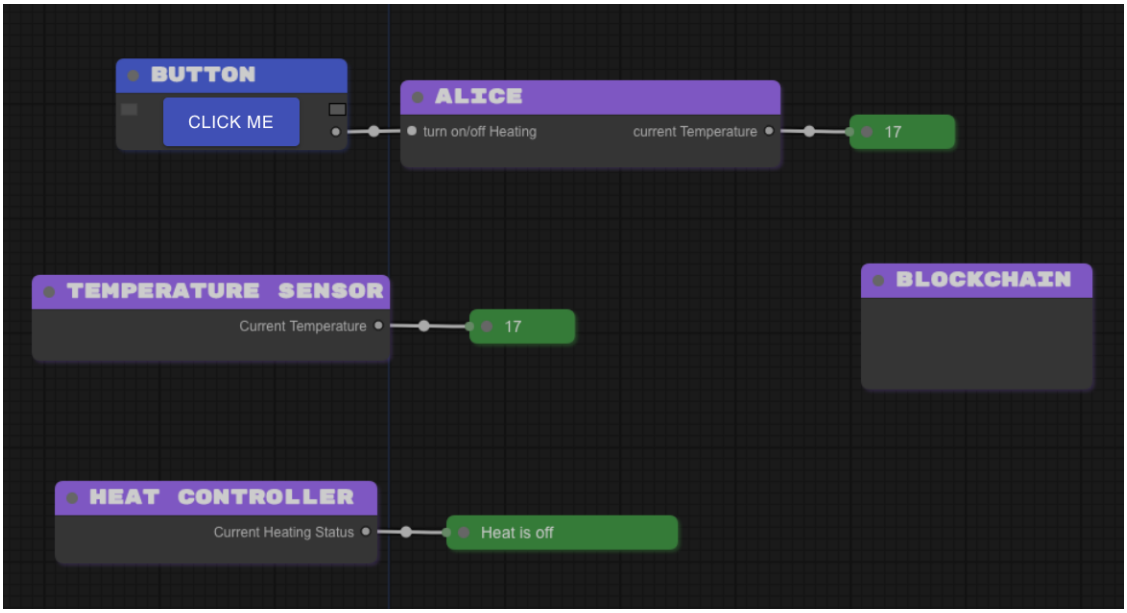


Figure 31 Structure of Ethereum-based Smart Home System Prototype (Screenshot)

The structure of the Ethereum-based smart home system is slightly different to the centralized, server-based smart home system, which implementation can be seen in Figure 31. The temperature sensor sends the current generated temperature continuously to the blockchain and its current temperature can be seen as output of the module. Also, the heat controller can be called by using a specific channel and commanded to turn the heating on or off depending on the status of the heating. The heating is off by default as shown in Figure 31. As soon as the controller receives a request the heating will change and turn on. Also, the current heating status can be seen from outside of the module due to the used output function. The blockchain has no input nor output functions because the temperature sent to the blockchain is sent

wirelessly via the publish function. This is why no input function is needed. There is also no output function because the only output the blockchain could offer is a list of all transactions but only the current temperature is important and the functionality for identifying the latest temperature is part of the user module. Finally, the user or Alice has compared to the other prototype only one input module and the same output module. The turn on/off heating input function is required for sending a request to the heat controller. The other function for getting all temperature values is not required here because Alice has access to the blockchain whereas the user of the other prototype must send a request to the server every time the list of temperatures is required.
In addition, an overview of all used publish and subscribe channels can be seen in Table 5 including which component sends or subscribes to it.

Table 5 Overview of all publish and subscribe channels and their sender and receiver components of Ethereum-based Smart Home System.

Channel	Sender	Receiver
blockchain	Temperature Sensor	Blockchain
controller-turnOnOffHeating	User	Heat Controller

4.2 CONDUCTION OF EXPERIMENT

After both subchapters describe the implementation of both prototypes, this subchapter deals with conducting the designed experiments including the execution of five different cyberattacks namely eavesdropping, traffic analysis, denial-of-service, man-in-the-middle, and masquerading attack once against both created prototypes.
The conduction of each cyberattack against both prototypes is described in the following subchapters.

4.2.1 EXECUTE CYBERATTACKS AGAINST CENTRALIZED, SERVER-BASED SMART HOME SYSTEM PROTOTYPE

The following subchapters describe the execution and results of the five defined cyberattacks executed against the centralized, server-based smart home system prototype.

4.2.1.1 EXECUTION OF EAVESDROPPING ATTACK AGAINST CENTRALIZED, SERVER-BASED SMART HOME SYSTEM PROTOTYPE

As already described, the purpose of an eavesdropping attack is to listen in on the attacked system without the system noticing. Since it is assumed that the interfaces of the system are known, there is one instance per interface that listens to it and stores its communication. To be able to track this as an attacker, there is a timer, which is shown in Figure 32. The purpose of this timer is to count the elapsed time from the beginning of the attack so that the messages can be allocated over time.

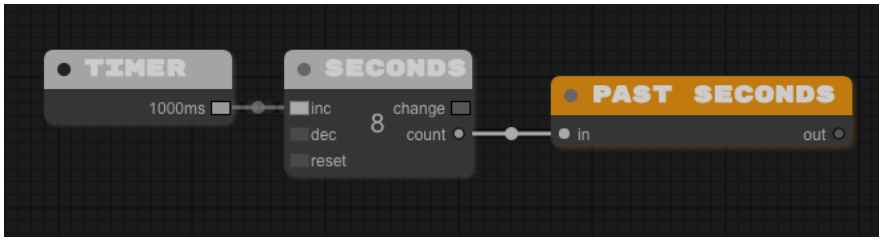


Figure 32 Timer of Eavesdropping Attack in centralized, server-based Smart Home System (Screenshot)

Therefore, a timer function is used which triggers an event every 1000ms or 1s, which in turn triggers a counter which is incremented by 1 after each event. This represents the elapsed time in seconds. The value is stored in a variable to be used in the different listening instances.

In an eavesdropping attack, each listening instance has the same structure and differs only in the interface it listens to. Figure 33 shows the listening instance on the "server-temperature" interface. It has a subscribe function that listens on "server-temperature". When this function receives a message, it increments a counter whose value is the input of a selector function. If the counter is an even number, then the selector selects input A, which has no value, and if the counter contains an uneven number, then the selector selects the elapsed time in seconds, which is created by the timer. This timestamp is merged with the content of the received message and a "to" value, which is the interface, into an object, which is finally added to a list as json. This list stores all received messages of the interface and could be sighted by the attacker. By inserting a delay after receiving the message, the counter is lowered by 1 and thus reset to the original state. This mechanism ensures that adding a new message to the list works.

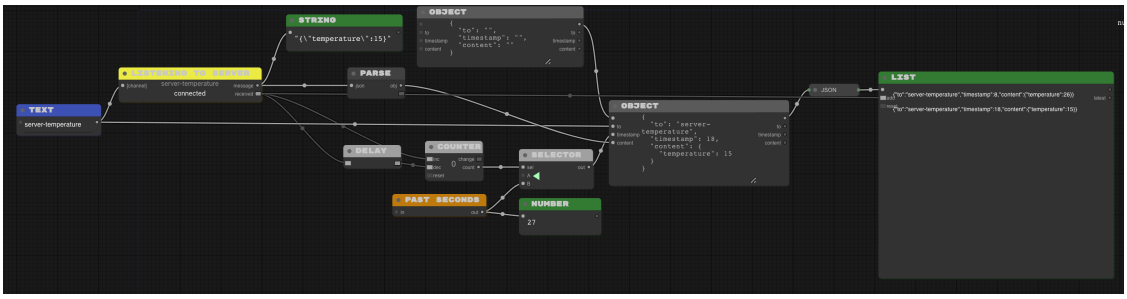


Figure 33 “server-temperature” channel Listening Instance of Eavesdropping Attack in centralized, server-based Smart Home System (Screenshot)

Of these listening instances, there is one for "controller-turnOnOffHeating", "server-temperature", "server-currentTemperatures" and "user-currentTemperatures" channel. For the experiment, the heating has been turned on and off by the user and the current temperature was queried twice by the user to determine that the functions continued to work without issue. The execution worked without problems and therefore does not seem to be affected by the eavesdropping attack.

The attack has been run for 100 seconds, and the information collected from the eavesdropping attack are summarized in the following table considering the recorded API and the collected information.

Table 6 Collected Information from Eavesdropping Attack against centralized, server-based Smart Home System

API	Collected Information
controller-turnOnOffHeating	{„to“:“controller-turnOnOffHeating“, „timestamp“:31, „content“:“Please turn Heating on/off!“}, {„to“:“controller-turnOnOffHeating“, „timestamp“:32, „content“:“Please turn Heating on/off!“}
server-temperature	{„to“:“server-temperature“, „timestamp“:8, „content“:{“temperature“:26}}, {„to“:“server-temperature“, „timestamp“:18, „content“:{“temperature“:15}}, {„to“:“server-temperature“, „timestamp“:28, „content“:{“temperature“:12}}, {„to“:“server-temperature“, „timestamp“:38, „content“:{“temperature“:15}}, {„to“:“server-temperature“, „timestamp“:48,

	„content“:{“temperature“:12}}, {„to“:“server-temperature“, „timestamp“:58, „content“:{“temperature“:30}}, {„to“:“server-temperature“, „timestamp“:68, „content“:{“temperature“:13}}, {„to“:“server-temperature“, „timestamp“:78, „content“:{“temperature“:13}}, {„to“:“server-temperature“, „timestamp“:88, „content“:{“temperature“:17}}, „to“:“server-temperature“, „timestamp“:98, „content“:{“temperature“:20}}
server-currentTemperatures	{„to“:“server-currentTemperatures“, „timestamp“:38, „content“:“Please send all Temperatures! “}, {„to“:“server-currentTemperatures“, „timestamp“:75, „content“:“Please send all Temperatures! “}
user-currentTemperatures	{„to“:“user-currentTemperatures“, „timestamp“:39, „content“:“[26,15,12]“}, {„to“:“user-currentTemperatures“, „timestamp“:76, „content“:“[26,15,12,15,12,30,13]“}

The attack successfully recorded the information sent between the individual instances and did not affect the prototype's function, nor could its actions be recorded or witnessed in any way by other instances. Therefore, the eavesdropping attack against the centralized, server-based smart home system prototype is considered successful.

4.2.1.2 EXECUTION OF TRAFFIC ANALYSIS ATTACK AGAINST CENTRALIZED, SERVER-BASED SMART HOME SYSTEM PROTOTYPE

In traffic analysis, an eavesdropping attack is executed, and its information is then analysed to discover sensitive data or functionalities of the system. Since the eavesdropping attack on the smart home system was successful, the results from that attack from Table 6 are used in the traffic analysis.

In general, it is noticeable that none of the messages sent are encrypted, so it can be assumed that the system operates without encryption. In addition, it could be

concluded from the names of the channels that there are possibly different parties in the system called controller, server, and user. These parties could have their own interfaces with which they can communicate according to their purpose and therefore the names of the channels are structured as they are.

Looking at the messages in the "server-temperature" channel, it is suspicious that this channel received a message exactly every 10s. Since the attack was executed for 100s and therefore 10 entries were recorded, it can be concluded that every 10s an instance sends a message to the channel. This would be further verifiable should the attack be performed repeatedly and for a longer period, but based on the available data, this can be assumed. Furthermore, the content of the message delivered every 10s is apparently a temperature. This temperature value does not seem to change according to any mathematical scheme and therefore appears to be random.

The information collected for the "controller-turnOnOffHeating" channel appears to be commanded to turn a heater off or on based on the name of the channel and the message. It is unclear who sent this message and whether there is a traceable rhythm since there are only two entries for the channel and the messages were sent to the channel one second apart. Therefore, it can only be concluded that the channel has something to do with switching on and off a heater.

The channel "server-currentTemperatures" sends as content the same message "Please send all Temperatures!". About this message and the name of the channel could be assumed that the channel is dealing with current temperatures. But what exactly this message possibly triggers or means is not entirely clear. However, it could possibly be about the information from the channel "server-temperature" but there is no exact clue for this. Also, the timing of the messages cannot be determined due to a lack of entries. It could be a coincidence that the first message arrived at 38s and the second message 37s later at 75s. It could also follow a mathematical scheme, but that is not clear from the data. Also, these messages do not seem to be related to any other messages in time.

The last channel is called "user-currentTemperatures" and their two messages were distinguished. Both messages send a list of numbers with them. As the name of the channel suggests, these numbers could be current temperatures. This is confirmed by the list of messages from the "server-temperature" channel because the sequence of numbers corresponds to the temperatures sent at that time. The first message to the "user-currentTemperatures" was sent at 39s and at that time already three messages were sent to the "server-temperature" channel, namely the temperatures 26, 15, and 12,

which were sent exactly in this order as a list at 39s to the "user-currentTemperatures" channel. The same is true for the second message. Therefore, it can be assumed that this channel receives the current lists of temperatures. It is noticeable that this channel received both messages exactly 1s after the message from the "server-currentTemperatures" channel. In addition, that also the channel names are pretty much the same, this seems to be no coincidence. Therefore, someone seems to have queried the server, which then writes to the user. Lastly, the messages from the "user-currentTemperatures" channel do not seem to have any temporal effect on the other channels.

In summary, there seem to be different parties in the system. There is a controller that can turn a heater on and off based on the sent messages, there is a server that receives a temperature every 10s and can be queried for the current temperatures. Following this message, the server sends the temperatures transmitted at that time to the user. It is unclear whether the user also sent the first request to the channel "server-currentTemperatures" and who can write messages to the controller and whether there are other parties in the system that just did not communicate with each other during the 100s attack. But nevertheless, based on the analysis of the messages, some information about the functioning of the system could be gathered that could be useful for further attacks such as the parties of the system or communication threads or functions of the system, and therefore the traffic analysis attack is considered successful.

4.2.1.3 EXECUTION OF DENIAL-OF-SERVICE ATTACK AGAINST CENTRALIZED, SERVER-BASED SMART HOME SYSTEM PROTOTYPE

For the denial-of-service attack, parts of the system must be limited in their usability for the user. The controller channel "controller-turnOnOffHeating" and the server channel "server-temperature" are attacked. For this purpose, many messages are sent to the respective channel. The implementation for sending messages to the "controller-turnOnOffHeating" channel can be seen in Figure 34. However, the implementation is the same for all attacked channels. A message with the content "denial of service" is sent to the respective channel every 10ms or 1s.

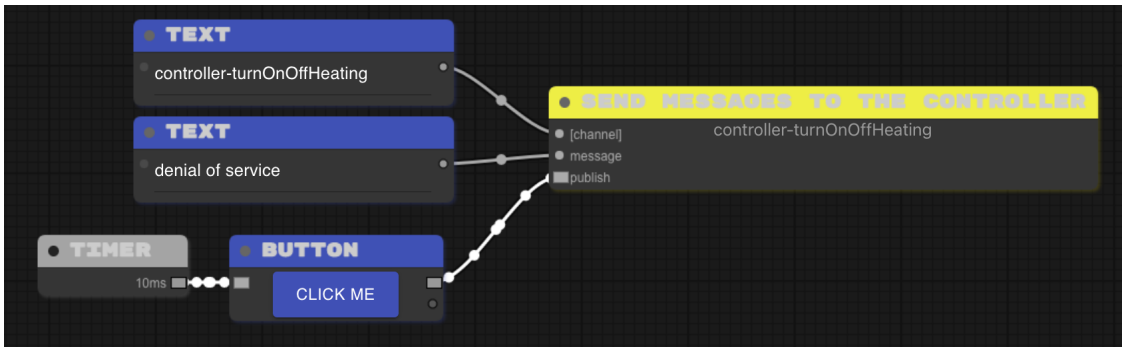


Figure 34 Denial-of-Service Attack sending messages to "controller-turnOnOffHeating" Channel against centralized, server-based Smart Home System (Screenshot)

Looking at the smart home system during the attack, it is noticeable that every second the heat controller changes its output from "Heating is off" to "Heating is on" and vice versa. When the user is triggered to turn the heat controller on or off, this request arrives at the heat controller. This can be seen by the fact that the current heating status stays on one status for a little while longer. Because every second a new message from the denial-of-service attack is sent to the heating controller, the user command is overwritten by the new command. Therefore, the heating controller is no longer controllable by the user.

When attacking the channel "server-temperature", it is noticeable that looking at the implementation, which can be seen in Figure 35, during the 100s execution time of the attack, a single temperature from the sensor got through to the server, although there should have been 9. Also noticeable is that the last message received on this channel has the text "denial of service" in it, which indicates that the messages of the denial-of-service attack arrived there.

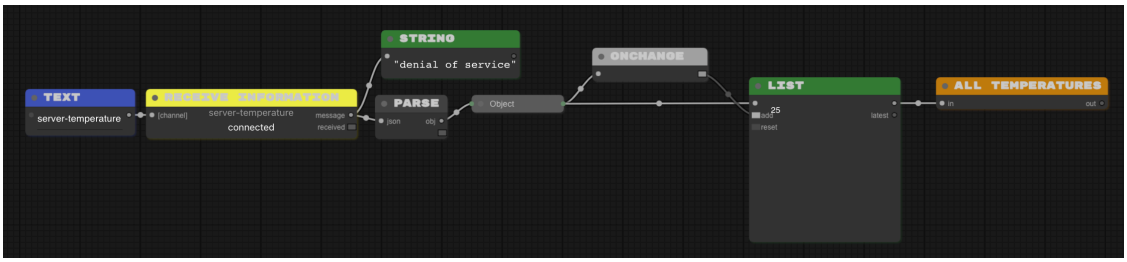


Figure 35 Implementation of "server-temperature" Channel during a Denial-of-Service Attack against centralized, server-based Smart Home System (Screenshot)

This has the effect that when Alice requests the current temperatures from the server, it receives a list with only one value and thus assumes that the current temperature is always 25, although it should change every 10s. This can be seen in Figure 36.

By sending many messages to the controller and the server, the user can only control the heat controller to a limited extent, and requesting the current temperatures returns an incorrect number of temperatures, but the function itself still works. Nevertheless, the attack was able to limit functionalities of the system and therefore the denial-of-service attack is considered successful. The uselessness of the temperature functionality becomes clear in the following figure, which shows a difference in the current temperature evaluated by Alice and the temperature sensor.

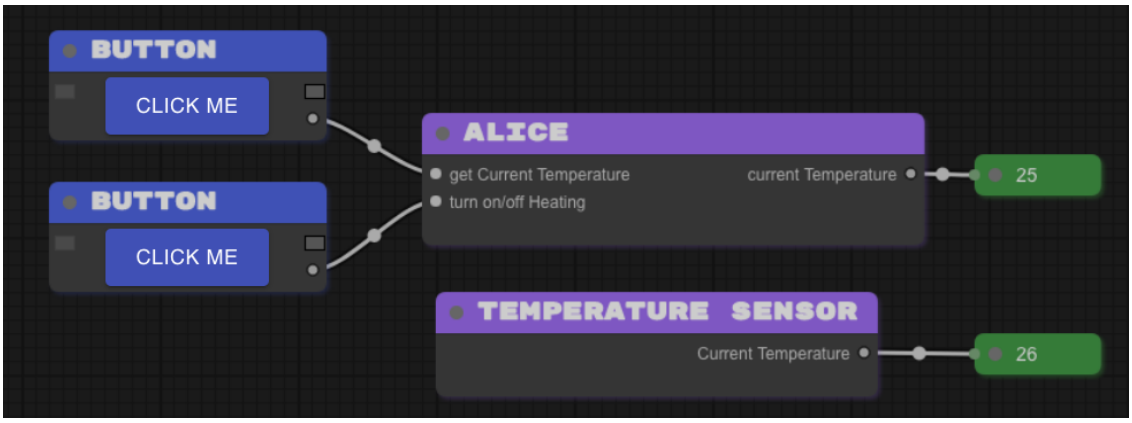


Figure 36 Comparison of Alice's current Temperature and the actual current Temperature from the Temperature Sensor during a Denial-of-Service Attack against a centralized, server-based Smart Home System (Screenshot)

4.2.1.4 EXECUTION OF MAN IN THE MIDDLE ATTACK AGAINST CENTRALIZED, SERVER-BASED SMART HOME SYSTEM PROTOTYPE

In the man-in-the-middle attack, the attacker must interpose himself between the communications of two parties to intercept and forward messages. Since each party sends against a specific channel, the attacker can listen to that channel. However, there is no possibility that a legitimate user of the system sends exclusively to the attacker and thus the attacker forwards the message because every sending and receiving of messages has a defined target, namely a unique channel. The attacker could just listen to the same channel of a user and then forward messages to other users as for example if a user requests the list of temperatures from the server and the server sends the list back to the user, then the attacker could listen to the message to the server and then also send a list of temperatures back to the user and since there is no verification of the received message from the user, the user would accept the message. But nevertheless, this means that the attacker did not interpose himself between two legitimate users of

the system and intercepted their message as it is defined for a man-in-the-middle attack. Therefore, the man-in-the-middle attack is considered unsuccessful.

4.2.1.5 EXECUTION OF MASQUERADING ATTACK AGAINST CENTRALIZED, SERVER-BASED SMART HOME SYSTEM PROTOTYPE

During the masquerading attack, the attacker sends the same messages to parties of the system that other users of the system would send. Therefore, an intruder module is created that has the same implementations as the other participants in the system to mimic its messages. Figure 37 shows the attacker's implementation of how to send a message to the heat controller. There are two input functions so that the sending of this message can be triggered from the outside and the content of the message can be customized from the outside, as well.

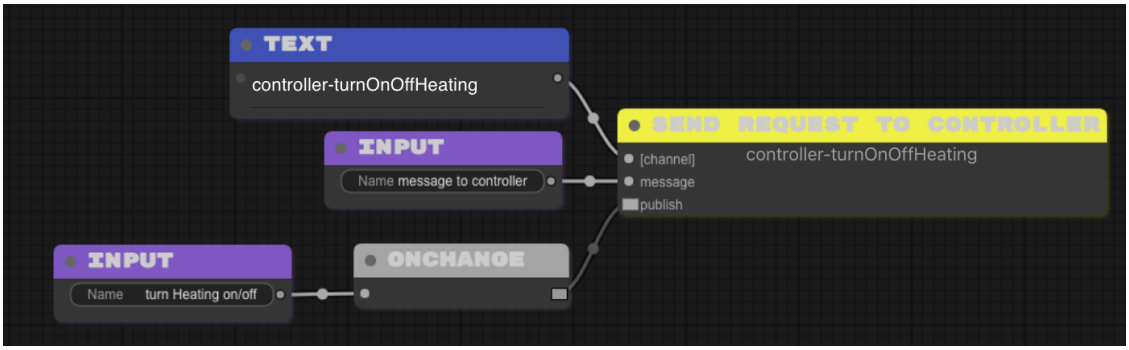


Figure 37 Implementation of the Attacker Sending a message to the Heat Controller during a Masquerading Attack against a centralized, server-based Smart Home System (Screenshot)

Similarly, the sending of temperatures to the server is imitated by sending messages to the "server-temperature" channel. The implementation of the attacker can be seen in Figure 38 and shows that this message can also be controlled from the outside using an input function and the sent message is also entered from the outside.

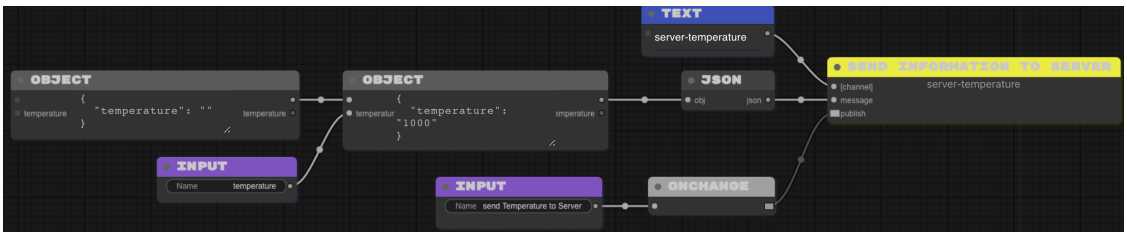


Figure 38 Implementation of the Attacker Sending the current temperature to the Server during a Masquerading Attack against a centralized, server-based Smart Home System (Screenshot)

Finally, there is communication from the user to the server to request the current temperatures. This communication is also mimicked by the attacker and can be seen in Figure 39. The implementation is once again the same as the implementation in the user module, but it can be triggered from outside and the message can also be changed from outside the attacker module.

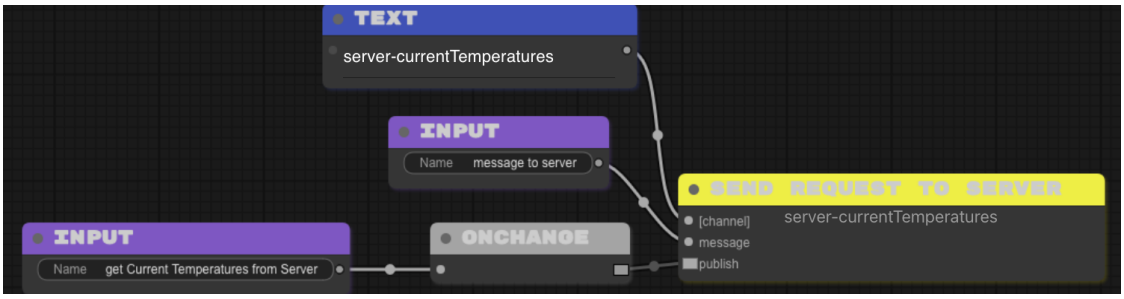


Figure 39 Implementation of the Attacker Sending a request to the „server-currentTemperature“ channel during a Masquerading Attack against a centralized, server-based Smart Home System (Screenshot)

Figure 40 shows the implementation of the "server-temperature" functionality in the server. There are multiple unrealistic temperature values in the list of messages. The temperature sensor can only generate values in the range of 10 to 30 due to the settings in the random function. Therefore, the four-digit values cannot come from the sensor and be injected by the masquerading attack. During the attack duration of 100s, two numbers were injected by the attacker, so as a result, 12 temperatures can be found in the server's list, although only 10 could have been present since the temperature sensor only generates a temperature every 10s.

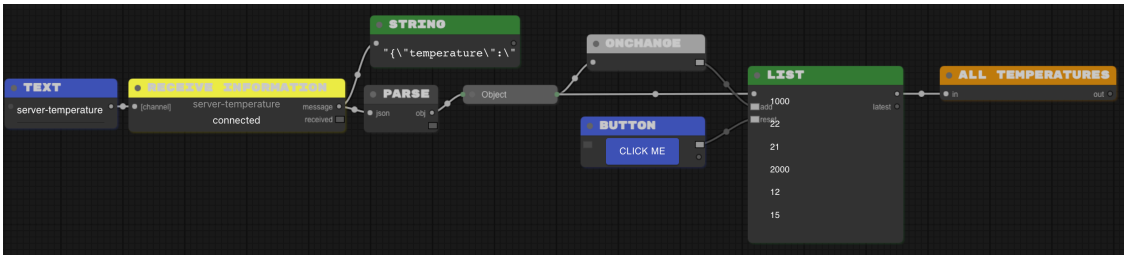


Figure 40 Implementation of "server-temperature" channel during the Masquerading Attack against centralized, server-based Smart Home System (Screenshot)

Operating the heat controller by the attacker also works without any problems. This can be seen in Figure 41, showing the implementation of the heat controller and that not only the output shows "Heat is on", although the heating is off by default, but also the send message "Do want I want!" does not correspond to the user-defined message "Please turn Heating on/off!”.

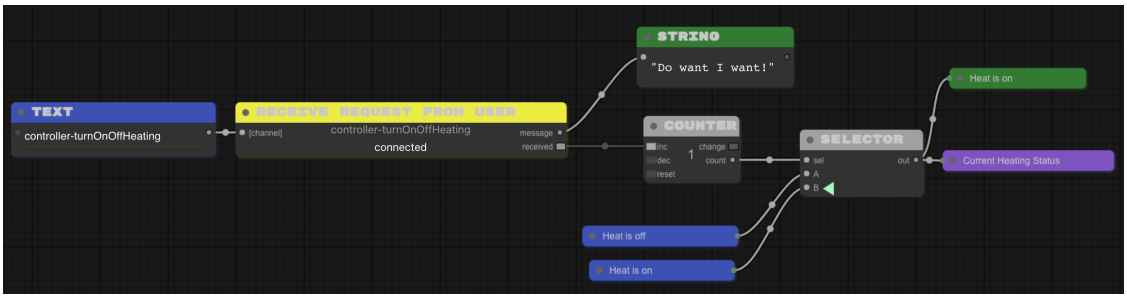


Figure 41 Implementation of the "controller-turnOnOffHeating" channel during the Masquerading Attack against centralized, server-based Smart Home System (Screenshot)

Finally, there is one interface that has not yet been addressed by the attack, the "server-currentTemperatures" channel. Sending a message to this channel only has the effect that the legitimate user gets the current list of temperatures and not the attacker, nevertheless, the attacker can address the interface without problems. That the attack worked can be seen in Figure 42, which shows the implementation of the "server-currentTemperatures" channel in the server. The last received message displayed in this figure shows "Give me the temperatures", which is not identical to the "Please send all Temperatures!" message from the legitimate user.

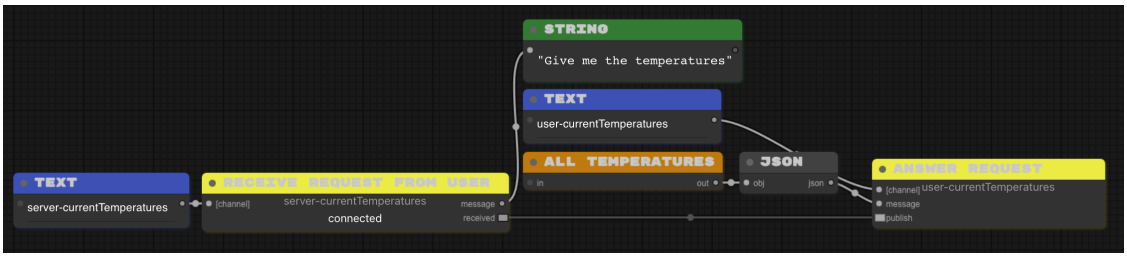


Figure 42 Implementation of the "server-currentTemperatures" channel during the Masquerading Attack against centralized, server-based Smart Home System (Screenshot)

In summary, the attacker managed to send messages to interfaces without any problems. This allowed him to operate the heating controller and send incorrect temperatures to the server. He could also trigger the server to forward the current list of temperatures. Unfortunately, the temperatures were sent to the legitimate user and not to the attacker, but nevertheless, the attacker was able to send messages with the privileges of other parties of the system, and therefore the attack is considered successful.

4.2.2 EXECUTE CYBERATTACKS AGAINST ETHEREUM-BASED SMART HOME SYSTEM PROTOTYPE

The previous subsection has demonstrated the execution of the designed experiments against the centralized, server-based smart home system prototype. In this subchapter, the same experiments are executed against the Ethereum-based smart home system prototype and its results are described.

4.2.2.1 EXECUTION OF EAVESDROPPING ATTACK AGAINST ETHEREUM-BASED SMART HOME SYSTEM PROTOTYPE

As already described, the purpose of an eavesdropping attack is to listen in on the attacked system without the system noticing. Since it is assumed that the interfaces of the system are known, there is one instance per interface that listens to it and stores its communication. To be able to track this as an attacker, there is a timer, which is shown in Figure 43. The timer is identical to the timer implemented for the eavesdropping attack against the centralized, server-based smart home system prototype. The purpose of this timer is to count the elapsed time from the beginning of the attack so that the messages can be allocated over time.

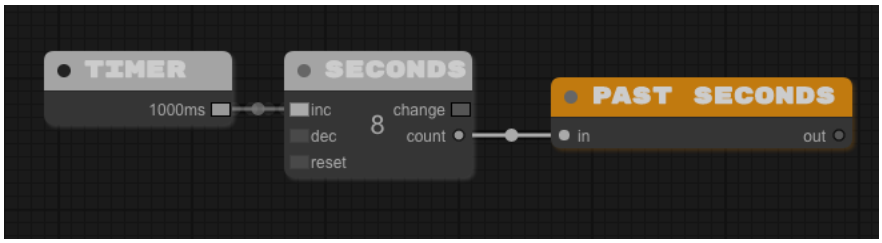


Figure 43 Timer of Eavesdropping Attack in Ethereum-based Smart Home System (Screenshot)

In an eavesdropping attack, each listening instance has the same structure and differs only in the interface it listens to. Figure 44 shows the listening instance on the "blockchain" interface. It has a subscribe function that listens on the "blockchain" channel. When this function receives a message, it increments a counter whose value is the input of a selector function. If the counter is an even number, then the selector selects input A, which has no value, and if the counter contains an uneven number, then the selector selects the elapsed time in seconds, which is created by the timer. This timestamp is merged with the content of the received message and a "to" value, which is the interface, into an object, which is finally added to a list as json. This list stores all received messages of the interface and could be sighted by the attacker. By

inserting a delay after receiving the message, the counter is lowered by 1 and thus reset to the original state. This mechanism ensures that adding a new message to the list works. The implementation of the Eavesdropping attack against the Ethereum-based smart home system is identical to the implementation for the centralized, server-based smart home system.

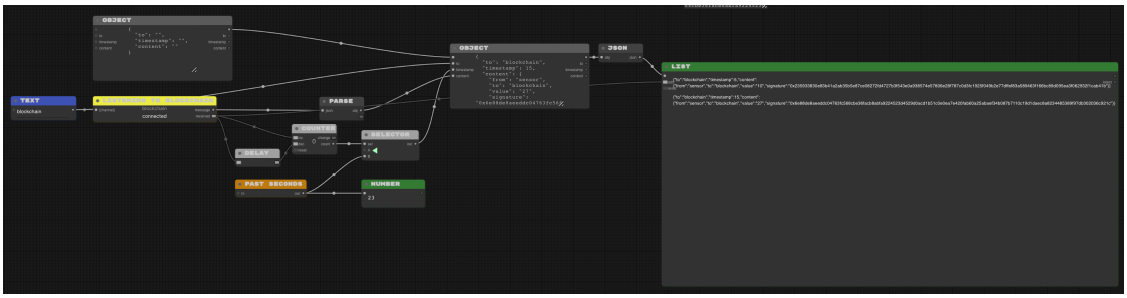


Figure 44 "blockchain" channel Listening Instance of Eavesdropping Attack in centralized, server-based Smart Home System (Screenshot)

There exists one listening instance for the "controller-turnOnOffHeating" channel and one for the "blockchain" channel.

For the experiment, the heating has been turned on and off by the user and the current temperature output of the user was checked during the experiment to ensure that all functionalities of the prototype are still working. The execution worked without problems and therefore does not seem to be affected by the eavesdropping attack.

The attack has been run for 100 seconds, and the information collected from the eavesdropping attack are summarized in the following table.

Table 7 Collected Information from Eavesdropping Attack against Ethereum-based Smart Home System

API	Collected Information
controller-turnOnOffHeating	{„to“:“controller-turnOnOffHeating“, „timestamp“:12, „content“:“Please turn Heating on/off!“}, {„to“:“controller-turnOnOffHeating“, „timestamp“:40, „content“:“Please turn Heating on/off!“}
blockchain	{„to“:“blockchain“, „timestamp“:5, „content“:“{"from": "sensor","to": "blockchain","value": "10","signature": "0x235933830e83b41a2ab35b5e87ce08272fd4727b3f543e0a938574e57806e28f787c0d3fc1925f049b2e77dffef83a589463f166bc89d095ea3f062932f1cab41b"}},

	{„to“:“blockchain“, „timestamp“:15, „content“:“{"from": "sensor","to": "blockchain","value": "27","signature": "0x6e88de8aeeddc04763fc568cbe36facb8abfa9224523d4529d0acd1b51c0e0ea7e420fab60a25abaef34b087b7110c19d1daec8a8234485389f97db302036c921c"}}, {„to“:“blockchain“, „timestamp“:25, „content“:“{"from": "sensor","to": "blockchain","value": "25","signature": "0x4835ad8bb98ace36971cfda44d6aa99be1efa4ecf9df7a5becc1a025c82edcf464d66e768df865d7b4f3471077e46a66adc7377bad9d7fe0c4dd622f639c672b1c"}}, {„to“:“blockchain“, „timestamp“:35, „content“:“{"from": "sensor","to": "blockchain","value": "26","signature": "0xb1c108ba3e77de4dfbb7d8dc40ba7692838b17e063f4df6e3cd7093754950410232981d8c975bc800395b12f5ffe09a857a00bd6e8e025f83a8fc34365ab98a41b"}}, {„to“:“blockchain“, „timestamp“:45, „content“:“{"from": "sensor","to": "blockchain","value": "12","signature": "0xf7e2b665d07b47065d1b5873c4fee965ad3a837f644884a730b7ec4cd92f6eee60f7f44a1d7280f81eacbecbcde01357f95d177bf1a3507d4dd8903bed783f611c"}}, {„to“:“blockchain“, „timestamp“:55, „content“:“{"from": "sensor","to": "blockchain","value": "30","signature": "0xfee75708b7ba2e2f312e73bb390183c0c0da0ff8a37e4adfea491ae852110e1f7aff5c6250a8fe125f99cb4924864d796582c37a4356f5c86137b9cf5db3c8841c"}}, {„to“:“blockchain“, „timestamp“:65, „content“:“{"from": "sensor","to": "blockchain","value": "12","signature": "0xf7e2b665d07b47065d1b5873c4fee965ad3a837f644884a730b7ec4cd92f6eee60f7f44a1d7280f81eacbecbcde01357f95d177bf1a3507d4dd8903bed783f611c"}}, {„to“:“blockchain“, „timestamp“:75, „content“:“{"from": "sensor","to": "blockchain","value": "19","signature": "0x105931bb791cbfa0c59295b8fba0a4e3492022af696a593

	e5d371b523d644b9065f3063f9c4c95b0aa605b27a474dd3ac b3226abcb581a1787e10385458eecfa1c"}}, {„to“:“blockchain“, „timestamp“:85, „content“:“{"from": "sensor","to": "blockchain","value": "13","signature": "0xe94d48eba7bdd6c11455d4804dd3a0ea204fedfbd8c6f33 66f732d1ef89c4d1542504c851929a6933ce7463163e1c1267 2c3f48e84f757705f118f59e1375ce61b"}}, {„to“:“blockchain“, „timestamp“:95, „content“:“{"from": "sensor","to": "blockchain","value": "12","signature": "0xf7e2b665d07b47065d1b5873c4fee965ad3a837f644884a 730b7ec4cd92f6eee60f7f44a1d7280f81eacbecbcde01357f9 5d177bf1a3507d4dd8903bed783f611c"}}

The attack successfully recorded information sent between the parties of the prototype and did not affect the prototype's function, nor could its actions be recorded or witnessed in any way by other instances. Therefore, the eavesdropping attack against the Ethereum-based smart home system prototype is considered successful.

4.2.2.2 EXECUTION OF TRAFFIC ANALYSIS ATTACK AGAINST ETHEREUM-BASED SMART HOME SYSTEM PROTOTYPE

In traffic analysis, an eavesdropping attack is executed, and its information is then analysed to discover sensitive data or functionalities of the system. Since the eavesdropping attack on the smart home system was successful, the results from that attack from Table 7 are used for the traffic analysis.

In general, it is noticeable that none of the sent messages are encrypted, but some messages contain a signature, so it seems that the system does not use encryption but a public-private key pair for signing messages. This is proven by recovering an address using each signature and each value from a message. All created addresses result in the same address: “0x6e4596ed1f35691ae9a4f7c5b833c707453556b5“. As signing a message proves that the sent message originates from the same user, it means that all the messages sent to the “blockchain” channel must be sent by the same user with the address “0x6e4596ed1f35691ae9a4f7c5b833c707453556b5“.

In addition, it can be concluded from the names of the channels that there are possibly parties in the system called controller and blockchain. Due to the name of the

controller channel and the received messages to that channel, it can be concluded that the controller's purpose is to turn the heating on or off. The purpose of the blockchain cannot be identified by the channel's name but the message sent to that channel seems to contain a value which is a number and according to the content of the messages sent to the blockchain are all messages coming from a sensor. This would be consistent with all messages appearing to be from the same sender based on the signature. That means that there is not only a blockchain and a controller but also a sensor part of the smart home system.

Looking at the messages in the "server-temperature" channel, it is suspicious that this channel received a message exactly every 10s. Since the attack was executed for 100s and therefore 10 entries were recorded, it can be concluded that every 10s an instance sends a message to the channel. This would be further verifiable should the attack be performed repeatedly and for a longer period, but based on the available data, this can be assumed. It is still not clear what the meaning of the number is, but it seems that the received values do not change according to any mathematical scheme and therefore appear to be random.

Furthermore, the communication between the two channels does not seem to be connected, as the messages to the controller channel appear to be sent randomly and the messages to the blockchain channel occur every 10s.

In summary, there seem to be different parties in the system. There is a controller that can turn a heater on and off based on the channel's name and the sent messages and there is a sensor sending every 10s one number to a blockchain. From this analysis, neither sensitive data could be extracted nor became it clear how exactly the system works. Nevertheless, according to the design of the experiment, it must have been only partially possible to perform traffic analysis and since it could be analysed that the system consists of at least three components and the blockchain component receives a value every 10s, the traffic analysis is considered successful.

4.2.2.3 EXECUTION OF DENIAL-OF-SERVICE ATTACK AGAINST ETHEREUM-BASED SMART HOME SYSTEM PROTOTYPE

For the denial-of-service attack, parts of the system must be limited in their usability for the user. The controller channel "controller-turnOnOffHeating" and the blockchain

channel "blockchain " are attacked. For this purpose, many messages are sent to the respective channel. The implementation for sending messages to the "controller-turnOnOffHeating" channel can be seen in Figure 45. However, the implementation is the same for all attacked channels. A message with the content "denial of service" is sent to the respective channel every 10ms or 1s.

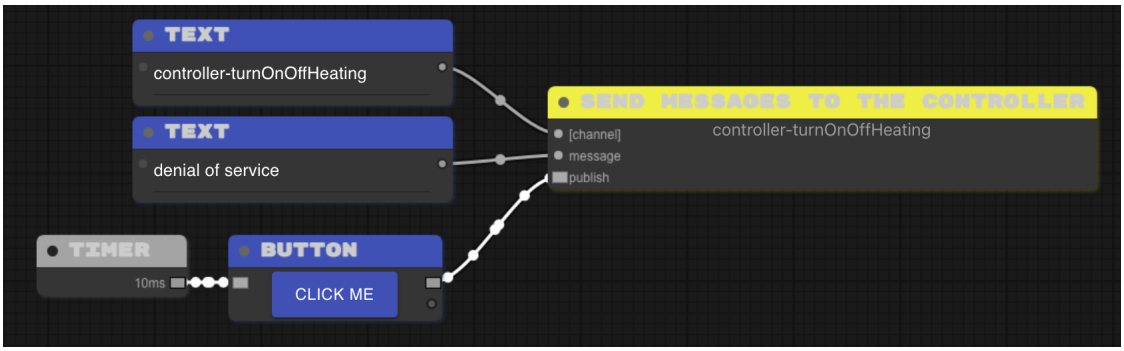


Figure 45 Denial-of-Service Attack sending messages to "controller-turnOnOffHeating" Channel against Ethereum-based Smart Home System (Screenshot)

Looking at the smart home system during the attack, it is noticeable that every second the heat controller changes its output from "Heating is off" to "Heating is on" and vice versa. When the user is triggered to turn the heat controller on or off, this request arrives at the heat controller. This can be seen by the fact that the current heating status stays on one status for a little while longer. Because every second a new message from the denial-of-service attack is sent to the heat controller, the user direct command is overwritten by the new command. Therefore, the heat controller is no longer controllable by the user.

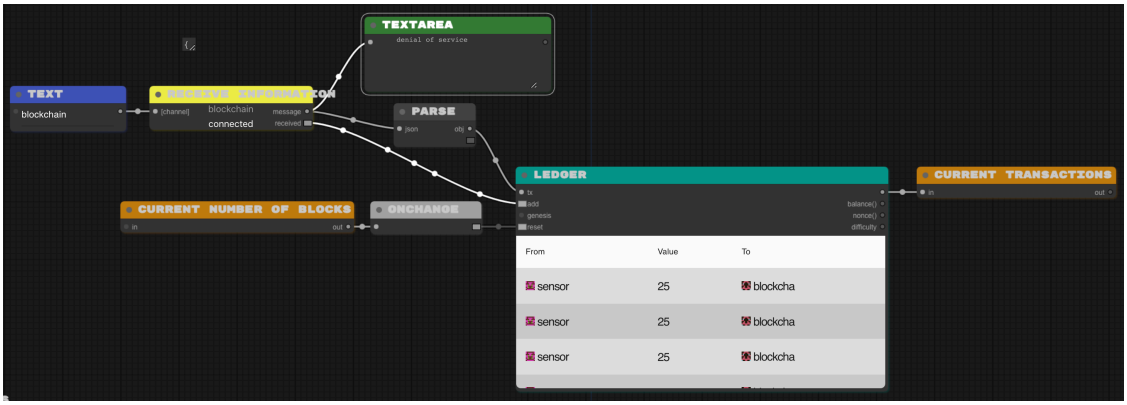


Figure 46 Implementation of "blockchain" Channel during a Denial-of-Service Attack against Ethereum-based Smart Home System (Screenshot)

Figure 46 shows the ledger and the corresponding blockchain channel during the denial-of-service attack. It is noticeable that the last received message is "denial of

service" and during execution it can be seen that the denial-of-service messages regularly arrive in the blockchain channel. Also visible is that the temperature sensor messages are still getting through to the ledger but for unknown reasons are being written to the ledger multiple times, as can also be seen in Figure 46 with a value of 25. Conversely, this means that the user can still read the current temperature from the blockchain even if it is duplicated multiple times.

By sending many messages to the controller and the blockchain, the user can only control the heat controller to a limited extent but can still examine the correct current temperature from the list of temperatures. Nevertheless, the design of the experiments defines that an attack is considered successful even though it only works partially on the system. The denial-of-service attack has limited the functionality of the heat controller and therefore the denial-of-service attack is considered successful.

4.2.2.4 EXECUTION OF MAN IN THE MIDDLE ATTACK ETHEREUM-BASED SMART HOME SYSTEM PROTOTYPE

In the man-in-the-middle attack, the attacker must interpose himself between the communications of two parties to intercept and forward messages. Since each party of the prototype sends against a specific channel, the attacker can listen to that channel. However, there is no possibility that a legitimate user of the system sends exclusively to the attacker and thus the attacker forwards the message because every sending and receiving of messages has a defined target, namely a unique channel. The attacker could just listen to the same channel of a user and then forward messages to other users as for example if a user requests the list of temperatures from the server and the server sends the list back to the user, then the attacker could listen to the message to the server and then also send a list of temperatures back to the user and since there is no verification of the received message from the user, the user would accept the message. But nevertheless, this means that the attacker did not interpose himself between two legitimate users of the system and intercepted their message as it is defined for a man-in-the-middle attack. Therefore, the man-in-the-middle attack is considered unsuccessful.

4.2.2.5 EXECUTION OF MASQUERADING ATTACK AGAINST ETHEREUM-BASED SMART HOME SYSTEM PROTOTYPE

During the masquerading attack, the attacker sends the same messages to parties of the system that other users of the system would send. Therefore, an intruder module is created that has the same implementations as the other participants in the system to mimic its messages. Figure 47 shows the attacker's implementation of how to send a message to the heat controller. There are two input functions so that the sending of this message can be triggered from the outside and the content of the message can be customized from the outside.

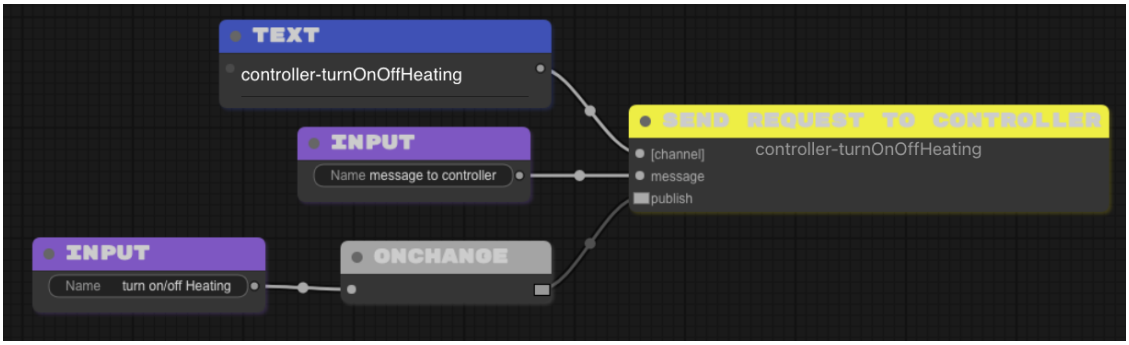


Figure 47 Implementation of the Attacker Sending a message to the Heat Controller during a Masquerading Attack against an Ethereum-based Smart Home System (Screenshot)

Similarly, the sending of temperatures to the server is imitated by sending messages to the "blockchain" channel. The implementation of the attacker can be seen in Figure 48 and shows that all values of that send object can be controlled from the outside using input functions.

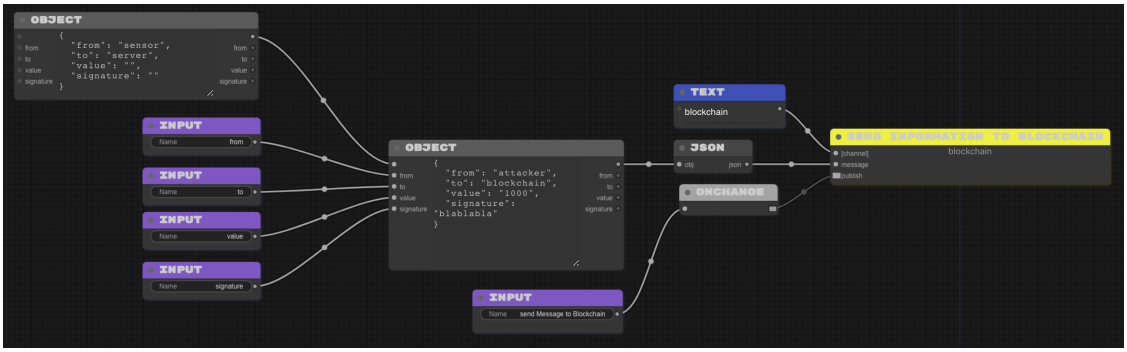


Figure 48 Implementation of the Attacker Sending the current temperature to the Ethereum blockchain during a Masquerading Attack against an Ethereum-based Smart Home System (Screenshot)

When the masquerading attack is executed by operating the heat controller with the attacker module, no problems occur. The attacker can operate the heat controller with the same privileges as the other legitimate users. The controller during the attack can

be seen in Figure 49, showing that not only the output shows "Heat is on", although the heating is off by default, but also the send message "Hello Heat Controller, here is the attacker" does not correspond to the user-defined message "Please turn Heating on/off!".

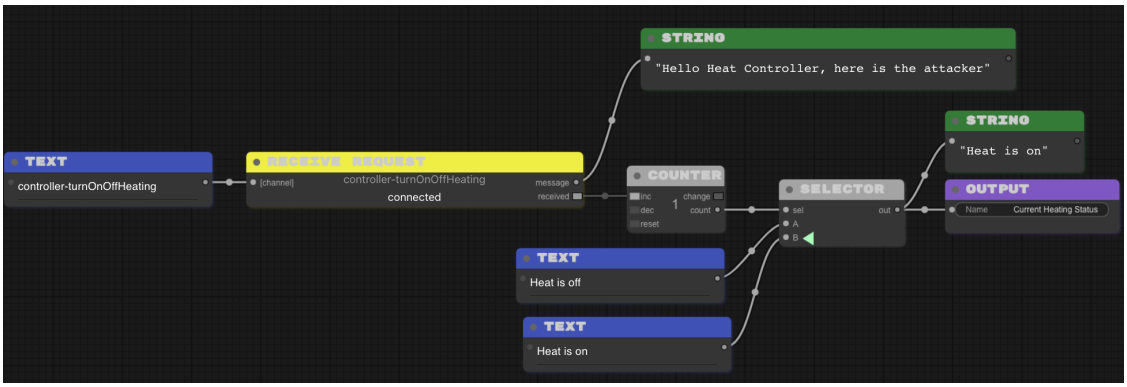


Figure 49 Implementation of the "controller-turnOnOffHeating" channel during the Masquerading Attack against Ethereum-based Smart Home System (Screenshot)

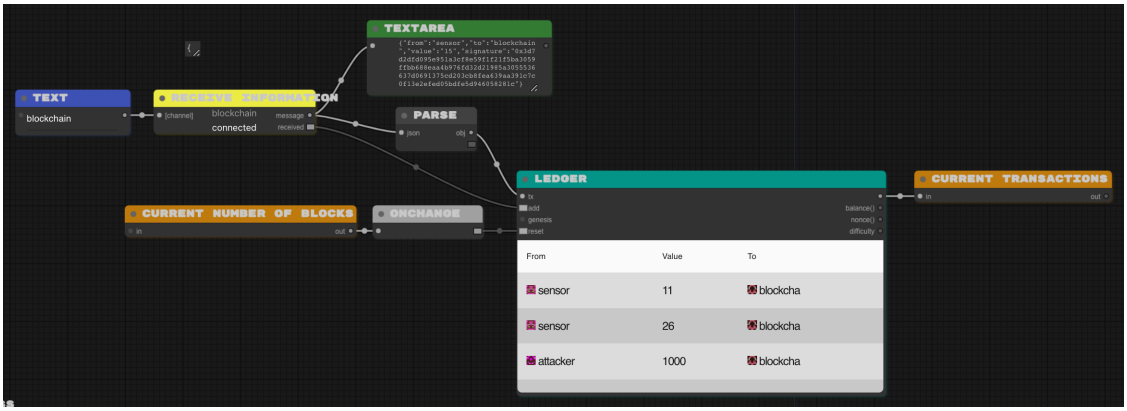


Figure 50 Implementation of "blockchain" channel during the Masquerading Attack against Ethereum-based Smart Home System (Screenshot)

Figure 50 shows the implementation of the "blockchain" functionality during the masquerading attack. There is an unrealistic temperature value in the ledger. The temperature sensor can only generate values in the range of 10 to 30 due to the settings in the random function and the “from” value is always set to “sensor”. Therefore, the 1000 value and “attacker” as “from” value cannot come from the sensor and be injected by the masquerading attack. During the attack duration of 100s, two numbers were injected by the attacker, so as a result, 12 temperatures can be found in the server's list, although only 10 could have been present since the temperature sensor only generates a temperature every 10s. But as the attacker does not have the private key of the temperature sensor and therefore cannot generate the same signature, the

implementation of the user will identify that the transaction is not originated from the temperature sensor and therefore will not accept the value as current temperature.

In summary, the attacker managed to send messages to the available interfaces without any problems. This allowed him to operate the heating controller and send incorrect temperatures to the blockchain. Nevertheless, the user can evaluate the messages signature and realize that the attacker's message is not send by the sensor and therefore ignores its content. However, the attacker has been able to operate the heat controller with the same privileges as other legitimate users, and therefore the masquerading attack against the Ethereum-based smart home system prototype is considered successful.

5 DISCUSSION AND CONCLUSION

The following subchapters give an overview of the whole research and draw a conclusion. Afterwards, the results are discussed, and the limitations of this work are shown as well as possible work for the future.

5.1 CONCLUSION

Compared to other studies that aim at improving the security of smart home systems by only focussing on improving the impacts of the used core technology, this research investigates the core of the security problem of smart home systems by comparing two technologies regarding their security. For this purpose, it was investigated whether an Ethereum-based smart home system can defend a higher number of different cyberattacks than a centralized, server-based smart home system.

Therefore, two smart home system prototypes were designed, one using a centralized server and the other an Ethereum blockchain. Both prototypes were designed and implemented in the same way to ensure comparability. Similarly, when selecting the five cyberattacks to test, the existing literature was considered to select passive and active cyberattacks which are considered as problematic for existing smart home systems and Ethereum blockchain. The experiments were also designed to ensure comparability between the two systems, so both prototypes are subjected to the same conditions.

Subsequently, the designs of both smart home systems were implemented digitally using the ETH.build application, and the experiments were conducted against these digital prototypes. The result of the experiment shows that both prototypes could not defend against eavesdropping, traffic analysis, denial-of-service, and masquerading attack, but could defend against a man-in-the-middle attack.

In summary, this research has addressed security issues in smart home systems. In contrast to previous studies, this research implemented a new approach proving that the level of security of smart home systems primarily depends on the used core technology. Two systems were studied, the classic smart home system with a centralized server and a new type of smart home system with an Ethereum blockchain. Both systems were examined for their security by executing five cyberattacks against both systems and examining their success. Many factors influenced the outcome of this

research and led to the result that an Ethereum-based smart home system cannot withstand more cyberattacks than a central server-based system. This result can only be understood considering the limitations of this work and should be further investigated in the future.

5.2 DISCUSSION

The research question of this study was to determine whether an Ethereum-based smart home system can defeat a higher number of different cyberattacks than a centralized, server-based smart home system. To this end, an Ethereum-based smart home system prototype and a centralized, server-based smart home system prototype were designed and developed. To verify the research question, five different cyberattacks were selected and one experiment per cyberattack was designed, which were finally executed against both prototypes. The result of the experiments revealed that neither prototype was successful in defending against four out of five cyberattacks, with only the man-in-the-middle attack being unsuccessful for both systems. This means for the research question that the Ethereum-based smart home system cannot withstand a higher number of different cyberattacks than a centralized, server-based smart home system.

Despite this, it was observed that the use of the Ethereum blockchain reduced vulnerability compared to the centralized smart home system. For instance, it was not possible for an attacker to imitate the communication of the temperature sensor in the Ethereum-based system due to the unique private key signature requirement for each message. However, as the experiment considered attacks successful even if they could infiltrate parts of the system, the masquerading attack and other attacks had to be considered successful in the Ethereum-based system and thus counted as unsuccessfully defeated.

5.3 LIMITATIONS

Several aspects exist that limit the validity of this study. As previously mentioned, the Ethereum-based smart home system prototype showed less vulnerability to attack compared to the centralized, server-based smart home system. However, the definition of a successful attack was established during the design phase to include attacks that only managed to attack parts of the system. This definition had a significant impact on

the outcome of this research. If the definition had been reversed to consider an attack as successfully defended if parts of the system could resist the attack, the result of this research would have been that the Ethereum-based smart home system can defend itself against a higher number of different cyberattacks than its centralized, server-based smart home system.

Furthermore, there are further limitations to the validity of the result due to constraints of the digital prototype created using the ETH.build software. This resulted in some variations in the implementation of the smart home system designs compared to what is described in the literature. For example, HTTP could not be used for wireless communication between the individual smart home system participants, as ETH.build does not support this. Instead, IPFS was utilized, which has similar functions to HTTP through publish and subscribe features. Additionally, the ETH.build software does not have an automatic block generation function for the Ethereum blockchain. Hence, the prototype was limited to multiple pre-coded blocks. Consequently, the blockchain mechanisms had to be built, such as that the next block is being filled as soon as the previous block has been mined. Lastly, for the experiments against the prototypes, it had to be assumed that the interfaces of the system are already known. This cannot be assumed for a real prototype and should have been done using a network scan, but it is not possible to implement such a scan in ETH.build. Furthermore, the network scan of a real prototype would be much more challenging if, for example, the prototype hangs in another network, is protected by a firewall, or the wireless communication might be via Bluetooth or Zigbee. These aspects could not be considered by ETH.build, so it had to be assumed that the interfaces of the system are known to the attackers.

Moreover, the validity of this research is limited by implementation details. This research did not implement additional security mechanisms such as authentication to make the used technologies comparable. Nevertheless, Ethereum has digital signatures, which are implemented in the prototype. However, these were only observed in the communication between the temperature sensor, Ethereum blockchain and the single user. In a comprehensive system, the digital signature would have had to be incorporated between Alice and the controller as well. But this decision would have limited the comparability of both smart home systems. Both decisions, for comparability or unity of both systems, are legitimate choices and have an impact on the validity of this research because, for example, the masquerading attack would not have worked on the Ethereum prototype if the communication between Alice and the

controller had also used digital signatures. Additionally, when implementing the Ethereum-based smart home system prototype, it was assumed that there is only one user and therefore only one miner in the system. If there are multiple miners in the system, then this also provides a greater vulnerability for the system, as other miners could mine a block that contains false information if the resources are greater. Lastly, it should be noted that also when designing the Ethereum-based prototype, it was decided to use a public blockchain. If the decision had been different and a private blockchain should have been used, then parts of the system would also have to be adapted because, firstly, not everyone can send to the blockchain and, secondly, there no longer needs to be a miner in the system. This could lead to fewer but also more potential vulnerabilities, which could possibly have led to a different outcome of the experiments.

Last, there are limitations to the validity of this research due to the limitations of the attacks. As already mentioned, ETH.build had to assume that the attackers already know the interfaces of the systems. However, it also had to be assumed that the attackers have access to the system, which may not be the case with a real prototype. Additionally, it is worth noting that the five cyberattacks were chosen based on the underlying literature. Nonetheless, other cyberattacks could have been selected which could also have potentially changed the outcome of this research.

Table 8 Limitations of this Research

Category of Limitations	Limitations
Design of Experiments	- Limitation due to the chosen definition of successful/unsuccessful cyberattacks.
Digital Prototype	- Limitation through implementation differences of the designed prototypes due to the usage of ETH.build application. - Limitations due to the assumption that the interfaces of both prototypes are known to the attacker. - Limitations due to implementation details.
Cyberattacks	- Limitations due to attackers having access to the system. - Limitations due to the chosen cyberattacks.

The described limitations are summarised in Table 8 visualizing the category of the limitations and all related and described limitations.

5.4 RECOMMENDATIONS FOR FUTURE WORK

There are various possibilities regarding the future. One option is to repeat the same study, but with different parameters for the prototypes and experiments to explore which technological factors contribute to improved security in smart home systems. It is also imaginable that additional security mechanisms such as authentication could be added to the design of the prototypes, or that the differences between wired and wireless communication between the components could be observed, or that Bluetooth or Zigbee could be used instead of the Internet as a means for direct communication. As mentioned in the previous chapter, more participants could be added to both systems to create a more realistic system for practical use at home. In addition to changes in the prototypes, experiments or cyberattacks could be adapted, such as performing multiple attacks simultaneously against the system or testing the security against social engineering attacks. This research could also serve as a basis for further development, with real prototypes being built and tested against the same cyberattacks instead of digital prototypes.

Also possible in the future is to use this research as a basis for further development. It could be further built upon and as a next step the same cyberattacks could be fired against the prototypes instead of digital prototypes, real prototypes could be built.

BIBLIOGRAPHY

Abbas, S., Farooq, M. S., Hwang, S. O., Khan, M. A., Khan, S., & Rehman, A. (2022). Blockchain-Based Smart Home Networks Security Empowered with Fused Machine Learning. Sensors, 22(12), 4522. https://doi.org/10.3390/s22124522

Abdullah, A. A. A., Abdullah, T. A. A., Ali, W., & Malebary, S. (2019). A Review of Cyber Security Challenges, Attacks and Solutions for Internet of Things Based Smart Home. IJCSNS International Journal of Computer Science and Network Security, 19, 139–146.

Ahmed, F. T., Hasan, S., Islam, M., Islam, S., & Pandit Antu, A. C. (2021). Prototype Design and Development of a Smart Home. 2021 International Conference on Engineering and Emerging Technologies (ICEET). https://doi.org/10.1109/iceet53442.2021.9659769

Albany, M., Alruwili, I., Alsahafi, E., & Elkhediri, S. (2022). A review: Secure Internet of thing System for Smart Houses. Procedia Computer Science, 201, 437–444. https://doi.org/10.1016/j.procs.2022.03.057

Alex, A., & Stephen, R. (2018). A Review on BlockChain Security. IOP Conference Series: Materials Science and Engineering, 396, 012030. https://doi.org/10.1088/1757-899x/396/1/012030

Ali, B., & Awad, A. (2018). Cyber and Physical Security Vulnerability Assessment for IoT-Based Smart Homes. Sensors, 18(3), 817. https://doi.org/10.3390/s18030817

Alphand, O., Amoretti, M., Claeys, T., Dall'Asta, S., Duda, A., Ferrari, G., Rousseau, F., Tourancheau, B., Veltri, L., & Zanichelli, F. (2018). IoTChain: A blockchain security architecture for the Internet of Things. 2018 IEEE Wireless Communications and Networking Conference (WCNC). https://doi.org/10.1109/wcnc.2018.8377385

Arabo, A. (2015). Cyber Security Challenges within the Connected Home Ecosystem Futures. Procedia Computer Science, 61, 227–232. https://doi.org/10.1016/j.procs.2015.09.201

Aung, Y. N., & Tantidham, T. (2017). Review of Ethereum: Smart home case study. *International Conference on Information Technology*. https://doi.org/10.1109/incit.2017.8257877

Benlamri, R., Khezr, S., Moniruzzaman, M., & Yassine, A. (2020). Blockchain for smart homes: Review of current trends and research challenges. Computers and Electrical Engineering, 83, 106585. https://doi.org/10.1016/j.compeleceng.2020.106585

Dorri, A., Jurdak, R., Kanhere, S. S., & Roulin, C. (2019). On the Activity Privacy of Blockchain for IoT. 2019 IEEE 44th Conference on Local Computer Networks (LCN). https://doi.org/10.1109/lcn44214.2019.8990819

Feher, N., Hölbl, M., & Zlatolas, L. N. (2022). Security Perception of IoT Devices in Smart Homes. Journal of Cybersecurity and Privacy, 2(1), 65– 74. https://doi.org/10.3390/jcp2010005

Gunawan, T. S., Ismail, N., Kartiwi, M., Mansor, H., Nordin, A. N., Yaldi, I. R. H., & Za'bah, N. F. (2017). Prototype Design of Smart Home System using Internet of Things. Indonesian Journal of Electrical Engineering and Computer Science, 7(1), 107. https://doi.org/10.11591/ijeecs.v7.i1.pp107-115

Hmeidi, I., Khamayseh, Y., Mardini, W., Shatnawi, F., & Yassein, M. B. (2019). Smart Home Is Not Smart Enough to Protect You - Protocols, Challenges and Open Issues. Procedia Computer Science, 160, 134–141. https://doi.org/10.1016/j.procs.2019.09.453

Hyeong-Ah, C., Lee, C., Kwanghee, C., & Zappaterra, L. (2014). Securing smart home: Technologies, security challenges, and security requirements. 2014 IEEE Conference on Communications and Network Security. https://doi.org/10.1109/cns.2014.6997467

Kim, T.-H., & Robles, R. J. (2010). A Review on Security in Smart Home Development. International Journal of Advanced Science and Technology, 15, 13–22.

Komninos, N., Lymberopoulos, D., & Mantas, G. (2011). Security in Smart Home Environment. Wireless Technologies for Ambient Assisted Living and Healthcare, 170–191. https://doi.org/10.4018/978-1-61520-805-0.ch010

Lee, Y., Park, J. H., Park, J. H. & Rathore, S. (2020). A blockchain-based smart home gateway architecture for preventing data forgery. Human-centric Computing and Information Sciences, 10(1). https://doi.org/10.1186/s13673-020-0214-5

McAuley, D., Piasecki, S., & Urquhart, L. (2021). Defence against the dark artefacts: Smart home cybercrimes and cybersecurity standards. Computer Law & Security Review, 42, 105542. https://doi.org/10.1016/j.clsr.2021.105542

Ozyilmaz, K. R., & Yurdakul, A. (2019). Designing a Blockchain-Based IoT With Ethereum, Swarm, and LoRa: The Software Solution to Create High Availability With Minimal Security Risks. IEEE Consumer Electronics Magazine, 8(2), 28–34. https://doi.org/10.1109/mce.2018.2880806

Patruni, M. R., & Saraswathi, P. (2022). Securing Internet of Things devices by enabling Ethereum blockchain using smart contracts. Building Services Engineering Research and Technology, 43(4), 473–484. https://doi.org/10.1177/01436244221078933

Sisavath, C., & Yu, L. (2021). Design and implementation of security system for smart home based on IOT technology. Procedia Computer Science, 183, 4–13. https://doi.org/10.1016/j.procs.2021.02.023