Heimdallr: A gatekeeper for IoT

Ayberk Aksoy

A Thesis
in
The Department
of
Computer Science
and
Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Computer Science at

Concordia University

Montréal, Québec, Canada

June 2019

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Ayberk Aksoy

Entitled: Heimdallr: A gatekeeper for IoT

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final Examining Committee:

______________________________	Chair
Aiman Hanna	
______________________________	Examiner
Joey Paquet	
______________________________	Examiner
Dhrubajyoti Goswami	
______________________________	Supervisor
Bipin C. Desai	

Approved by ____

Chair of Department or Graduate Program Director

_____________ 2019 ____

Dean of Faculty

ABSTRACT

Heimdallr[1]: A gatekeeper for IoT

Ayberk Aksoy
CSE, Concordia University, Montreal Canada
a_aksoy@encs.concordia.ca

In today's wired and interconnected world, a sheer number of devices are now able to connect to the Internet and expose the data generated by user inputs or the devices' built-in sensors. These growing numbers of Internet of Things (IoT) devices are called smart and they range from mobile phones, smart televisions, IP cameras, household and industrial appliances, to Wi-Fi thermostats and thermometers. The reason for the avalanche of IoT devices is their convenience and remote accessibility over the traditional versions. However, as with other technological breakthroughs, IoT has major issues regarding the security of access and control of the data generated and hence privacy.

To address these issues, we propose in this thesis a system-based approach: the system consists of two parts to monitor users' IoT devices. The first aspect of this system is a firewall monitoring and controlling the incoming and outgoing traffic to and from the IoT devices which are connected to the Internet via a new generation of modem/router called Heimdallr. The second aspect is to store locally in this modem/router the user's IoT related data and allow a secure interaction by only the "owner" user with these data and the IoT devices. A new concept is introduced in the system to ensure that any update to the IoT software is verified and certified by a central not-for-profit organization. Heimdallr would not allow any update to the IoT's software to be made unless the update has been certified by this certification agency. The certification agency has a role similar to CSA [1] or UL [2] organizations which provide testing, inspection and certification services and are involved in setting standards.

[1] Heimdallr, in the Norse mythology, is the gods' watchman having acute hearing and eyesight.

ACKNOWLEDGEMENTS

I would like to first thank my parents, Ayla Aksoy and Aydın Aksoy for making everything in my life possible. Without their support I wouldn't be able to achieve any of this, thus, I have all the right to consider myself the luckiest person for being their child. And I am proud to dedicate my achievements and success to them.

Secondly, I cannot thank Dr. Bipin C. Desai enough for being not only my supervisor but also a second father to me being an international student far away from my home and family. His passion and knowledge on the topic has always been my guide throughout this research. He was the one who opened my eyes to the problems with IoT systems and he was also the one who gave me the first spark to our proposed solution in this project. Beside of the academics, he was always there for me whenever I needed any kind of support or whenever I felt lonely. It is never going to be enough how many times I say it but I was incredibly lucky to be your thesis student, thank you Professor!

I would also like to thank my uncle Cihan Aksoy and my cousin Rauf Alp Denktaş for supporting me both morally and financially not only throughout this work but also whenever I need them. Likewise, I would like to express my deepest appreciation to my grandmothers Seniha Evirgen and Nuran Aksoy, and my grandfathers Hasan Evirgen and Hicri Aksoy for all of their support and effort which makes me who I am today. Additionally, I owe a big thank you to rest of my cousins Pınar Denktaş, Can Denktaş, Aslı Evren Delilbaşı, and Derya Ülker Kıpçak as well as my aunts Mesude Denktaş, Fatma Mesrure Ülker, and Ayşe Ayfer Akalın, and rest of my family.

For his huge support in my success, I would like to express my sincere gratitude to the Executive Director of the International Energy Agency, Fatih Birol.

In addition, I would like to thank Dr. Aiman Hanna for his guidance in computer networks and security, Dr. Joey Paquet and Dr. Dhrubajyoti Goswami for their valuable feedbacks which helped me improve the final version of this thesis. I would also like to thank my program advisor Halina Monkiewicz for all of her help and support throughout my education in Concordia University.

Finally, a special thanks to all my friends who are always with me on my both worst and best days. I am so glad to have you!

Table of Contents

List of Figures

1 Introduction

The objective of this project is to introduce the current technology used for Internet of Things (IoT), a rapidly growing market, and the privacy and security issues associated with their use. Chapter 2 discusses these issues of IoT technology and some of the methods to address them and their suitability for actual usage in a private environment. Chapter 3 and 4 propose a novel solution called Heimdallr along with its high and low level architecture and design decisions. The experimentation of this, using a proof of concept simulator is given in Chapter 5 with its proposed hardware implementation, Chapter 6 is a conclusion and future work.

1.1 Introduction to IoT

Internet of Things, commonly called IoT is one of the most popular and ubiquitous invention among the newest technologies of the early 21st century. The name *"Internet of Things"* likely appeared after the title of a presentation made by Kevin Ashton at Procter & Gamble in 1999 and it has being used since [3]. In the name Internet of Things, "*Things"* implies the devices that have the ability to connect to the Internet. And the concept of creating a network of those devices by connecting them to each other and allowing them to use online services over the Internet is called the IoT. In other words, "The IoT is a giant network of connected things and people – all of which collect and share data about the way they are used and about the environment around them" [4].

The first IoT device was introduced in 1982 where a modified Coke vending machine which was able to report whether the drinks in its inventory were cold or not, was presented at Carnegie Mellon University [5]. Like many of today's smart devices, an Internet of Things (IoT) device claims to be smart. These smart devices replace the function of some traditional device with the ability of not only replacing the function, perhaps using other technology and built-in sensors, but adding a feature to have an internet connection. Using those sensors and internet connection, it collects data and sends them to an Internet of Things platform, sometimes called as a cloud which integrates data from different devices and applies analytics to produce a valuable information to be used by an application built to address a specific need [4]. This need can be related to medical fields, childcare or kitchen appliances and many more. The internet connection feature is also needed to connect the device to some transmission device, usually a cell phone (another smart device), or a controller connected to the Internet and it is essentially a “black-box” as far as the consumer is concerned. The cell phone or the “black-box” is used to complete the connection of the device to a server which is usually operated and controlled by the manufacturer of the IoT device. This connection is used to transmit data from and to the IoT device and provide remote access to its user, e.g. IP cameras. The smart in these devices replaces the mental or temporary recording of the status of some data: the app and its infrastructure takes over this operation adding some convenience to ‘justify’ the higher cost and effort of replacing the old and tried traditional device. Today, with the current technology, it is not that unusual to see the replacement of a traditional device with its IoT version which ultimately makes it more practical and convenient for daily use.

1.2 State of the Art

Usage of IoT devices is proliferating as the required technology becomes widely available and cheaper [6]. Currently (2019) there are 7 billion IoT devices being used worldwide excluding the smartphones, tablets and laptops which are in fact IoT devices as well. According to an article published by IoT Analytics, this number is expected to grow to 10 billion by 2020 and 22 billion by 2025 [7]. Moreover, it was reported in October 2016 by the research firm Venture Scanner that there were 1,428 IoT startup companies across 48 countries with a total value of 25 billion US dollars [8]. In 2018, IoT Analytics updated that number of global market value for IoT to 151 billion US dollars with an increase of 37% from 2017 and added that due to the market acceleration for IoT, revisions done on the estimates show an expected total market value of 1,567 billion US dollars by 2025 [7]. The very reason behind this increase is the decrease in cost of the technology, practicality of its usage, and the quality of the job done using those IoT devices compared to their traditional versions, therefore, the increase in demand.

We can see the incursion of IoT in many fields. An essential one of which is healthcare. For instance, a real-time monitoring of the condition of patients via an IoT device connected to a smartphone app can save lives in case of an emergency like heart failure, diabetes, asthma attacks, etc. by collecting and sending the critical information to a physician without any delay (provided someone is monitoring the alarm and reacts to it). Moreover, according to the study conducted by the Center of Connected Health Policy, when it comes to heart failure patients, there was a 50% reduction in 30-day readmission rate thanks to the remote patient monitoring [9]. Another example of IoT usage leads to smarter agriculture. Farmers now use IoT devices to track microclimates across cropland or to monitor humidity and temperature changes to act accordingly with their goods in order to eliminate any waste. Also, "According to The Nature Conservancy, such precision agriculture can enable farmers to cut water and fertilizer use by up to 40 percent, without reducing yields" [10]. Aside from those examples many other applications of IoT can be seen today in smart environments (e.g. smart homes, cities and cars), domestic (e.g. smart thermostats) and industrial (e.g. agricultural sectors) applications, security, emergency, logistics and transport fields [11].

When it comes to development of an IoT device there are four main areas involved as Forbes Insights says; hardware and software, edge computing, network connectivity, and data management [12].

- ***Hardware and software*** are the building blocks of an IoT device. Each device must compose of a hardware usually containing built-in sensors, a processing unit, and a power source. There are also some devices which fall in IoT category but don't have any processing unit or a power source of its own. An example for that could be a passive RFID[2] tag which extracts all of the power it needs to be functional from the carrier wave sent by the reader. The most common application of those tags are the systems that can detect the presence or absence of the tag and progress accordingly [13]. In case of a more complex IoT device, such as a smart thermostat, those devices requires a power unit to power up their processing unit which produces some valuable information from the data gathered through any build-in sensors. A smart thermostat would gather the room temperature through its type of temperature sensor and with the help of a software, compares it with the set (desired) value and then accordingly signal the cooler or the heater.
- ***Edge computing*** is the ability to do any kind of computation and decision done in the IoT device right after the data gathered without sending it to a server or a cloud. This is achieved by running a software on a processing unit indicated above. This idea mainly "helps improve speed and reduce network bandwidth" [12].

[2] Radio frequency identification system (RFID) is an automatic technology and aids machines or computers to identify objects, record metadata or control individual target through radio waves [89].

- ***Network connectivity*** is required to send and receive data between the user, device, and the server. For IoT devices the preferred type of medium for communication is wireless rather than wired. This is mainly because, even though wired communication is more viable and safer, its disadvantages on mobility and installation cost prompt IoT device makers to prefer an effective and low cost alternative, a wireless medium [11]. Today some of the most used wireless technologies are LPWANs[3], Cellular (3G, 4G, 5G[4]), Bluetooth and Bluetooth Low Energy (BLE), Wi-Fi (IEEE 802.11a/b/g/n) and Wi-Fi HaLow[5] (IEEE 802.11ah), RFID, and ZigBee[6] and other mesh protocols like Z-Wave and Thread[7] [14]. Almost all of the IoT systems being used today have the ability to connect to the internet using one those wireless technologies. Without a network connectivity an IoT device cannot benefit from any cloud service or provide remote access to its users and thus cannot fulfill its purpose properly. In further sections, this paper will talk about what kind of problems arise from having such a feature.
- ***Data management*** is another essential part of the IoT development. Some data that is collected through sensors of the IoT device may require more complex processing and therefore the help of cloud computing. In such a case, the dramatic increase in number of IoT devices and having generally a shorter time span for processing brings up big challenges on how to handle, process, and store the data sent to the cloud [12] [15]. Moreover, when it comes to personal data, security and privacy becomes a matter that cannot be overlooked by the IoT device maker. Unfortunately, in June 2017 it has already been reported by Newsweek that 48% of all IoT companies have experienced at least one security breach and the cost of a single breach was over 20 million US dollars for large firms [16]. In this project, a solution will be proposed to that essential problem of data privacy.

[3] A low-power wide-area network (LPWAN) is a type of wireless telecommunication wide area network for long range communications at a low bit rate among connected objects [85] [86].
[4] 5G is the fifth generation cellular mobile communication technology. Anticipated to be available in 2019.
[5] IEEE 802.11ah is a new Wi-Fi draft for sub-1GHz communications, aiming to address the major challenges of the Internet of Things (IoT) [90].
[6] ZigBee is an IEEE 802.15.4-based specification developed to enable creation of personal area networks with low-power IoT devices.
[7] Thread is an IPv6-based, low-power mesh protocol used for IoT products.

1.3 Conclusion

IoT is expanding in every domain of our lives and most or maybe even all of them will inevitably replace their older traditional versions in the near future. As though they are attractive, not everything is perfect and flawless about IoT devices. And despite this fact, there are still no good standardization when it comes to IoT device production.

These smart devices mostly require Internet connection to send and receive the essential data in order to function properly. However, with the lack of standards and regulations in IoT domains, users often end up losing the control over their sensitive and personal data, and thus, left with two choice. One is to not use these practical and highly efficient smart devices, or to give up on their privacy. Since none of those options are in favor of the user, in this thesis, we will first analyze these privacy issues in detail and then propose a third option in users' favor.

2 Background

Some of the core background information needed for a better understanding of the rest of the project will be provided in this section. First, the problems brought by the IoT devices and the device makers on data privacy will be explained in detail under the title of *Problem Definition*. The following section will discuss what and how competent the currently proposed solutions are in solving the mentioned issues.

2.1 Problem Definition

Many issues with IoT devices and networks are being noticed today: this reminds one of the days of Wild West when entire indigenous communities were wiped out by the advancing hordes of immigrant settlers with guns and artillery. Personal privacy is perhaps the most important victim of this digital age Wild West massacre.

The first weakness in IoT devices is that the development of their software is usually rushed by the organization introducing these devices with apparent lack of thought about security and privacy. Even in applications where the human life may be in jeopardy, developers can rush and overlook what must be or have been done. In case of an IoT device, nothing different should be expected. A glaring example of a product rushed through is illustrated by the design flaws, not only in the hardware but also the software of Boeing 737 Max which led to two fatal crashes. The reason for the rush was to catch up with competition [17].

The second issue is the lack of safeguard of the user data which is made available, for profit, to third parties. Hence the data subjects (original generators of the data) lose control of their data which exposes them to uninvited marketing and manipulation. All small incremental data, when aggregated by various players of this digital age gives what we call big data. Big data and the governmental *laissez-faire* attitude has led the exploitation of this data and has given rise to mammoth new so called tech companies. One of the most obvious and profitable way of exploiting user data collected and aggregated from users is to enable the marketing of targeted ads. For instance, the annual advertising revenue of one of these tech giants for 2018 exceeded 110 billion US dollars forming the major percent of this company's total revenues in that year of over 130 billion US dollars [18] [19]. Decades of research in marketing has concluded that rather than global publicity, targeted ads are what make the difference. In order to gain such data leverage the company keeps track of all the users of their ‘free’ services –along with their actions, sites visited, text communications and choices– and analyzes them against past usage and create profiles which guide their software to choose the most pertinent tailored targeted ads. However, they don’t stop there and this is exactly where the problem arises.

In order to increase the profit, these companies with big data sell user profiles to other companies who in turn can do the same. For instance, Facebook allows personal data to be used by paying third parties as Mark Zuckerberg revealed during his testimony before the Congress on April 10, 2018. During the testimony it was revealed that Aleksandr Kogan, a onetime lecturer at Cambridge University, had developed a survey application on Facebook platform and using the access given by Facebook, was able to assemble the personal data of all friends of any user who took the survey. Access to the friends data without the consent of these friends, was a feature of Facebook and any of the thousands of applications developed for the platform had access to the users’ and their friends data [20] [21].

Similar stories have been heard about other big tech companies as well. For instance, Google collects its users’ data and sells them to other companies without data owners’ awareness. As Douglas MacMillan stated in one of his articles in the Wall Street Journal (WSJ), Google let hundreds of outside developers use the data scanned from the inbox of their users who have signed up for ‘free’ email-based services [22]. Furthermore, it is reported that Google allows these outside developers to share what they collect with other third parties. In a letter from Susan Molinari, former vice president for public policy at Google, “Developers may share data with third parties so long as they are transparent with the users about how they are using the data,” she wrote, according to a WSJ article on September 20, 2018 [23]. It is not clear what this transparency means and/or allows; and whether the users are notified.

Among all other big companies with billions of users’ personal data, Amazon stands out the most when it comes to selling targeted ads, even though it has only around ten percent of Google or Facebook’s big data. Since advertisers value the most accurate information they can get on what people actually buy and what they are likely to buy soon, this can be gleaned from the Amazon marketing platform. Knowing this, according to another report from WSJ, even their employees leak data for cash rewards [24]. It appears that these tech mammoths are not living up to their own privacy policy which they can change without any oversight.

These reports were based on some incidents related to some services that users access using an internet browser. Now one may start to wonder, how much data we are giving away by just buying IoT devices and bringing them into our homes. Amazon Echo and Google Home and all the IoT devices that can be connected to them are such examples. Every word that the users say may not be sent to the cloud by those devices but they are always listening to be able to recognize their wake word and this can lead to some serious privacy breaches. Furthermore, Amazon Echo and Google Home are legitimate devices made by world giants who have so to speak strict privacy policies. But what about the other IoT devices, some are start-ups, which are rushed to market for profit and lacks safeguards regarding user privacy and data security? Do these companies which have access to user profiles have any policies or ethical code and how are they regulated?

Another example of IoT replacing traditional devices is a smart body thermometer. The traditional mercury in a glass tube clinical thermometer, was invented in 1724 by Daniel Gabriel Fahrenheit. It has been used since [25]. It needs no battery and is ready for use even after years of non-use; the only tricky part is to shake down the mercury to reset it. This classic clinical thermometer has been replaced by a digital version which requires a battery needing regular replacement. Both these are now being displaced by a smart body temperature thermometer; one by Kinsa Company is a case in point. These internet-connected thermometers replacing the traditional mercury column in glass versions and even the ordinary digital ones, are now in more than 500,000 households [26]. These thermometers send the data to an application in a cellphone to track the temperature; however, the data is also sent to the device manufacturing company. During the flu season of 2018 the Clorox Company paid to license information from Kinsa, according to NYT [27]. Kinsa also sells this data to other companies under the name of Kinsa Insights to be used for target advertising [27].

This is only one such example of IoT usage and how the users lose control over their personal data and privacy. Perhaps, losing control over the private data which consists of the user's first name, family name and email address might not be an issue to some extent, however, if this data gives away information about user's daily routine, this can lead to some serious unwanted consequences. Smart thermostats which allow users to see and alter the current temperature of their houses remotely, carries such data. As an example, if a smart thermostat company provides this user data to paying third parties or an overlooked hardware or software vulnerability of the thermostat gets discovered by a malevolent individual, a misuse of that data could reveal the absence of the house owner at low temperatures. Those kind of contemporary issues in the IoT sector show the importance of privacy, security and the users having control over their own data.

2.2 Proposed Solutions

After a thorough research which we will be discussing in detail in this section, we concluded that there isn't any direct and complete solution to current data privacy issues caused by the IoT devices and the IoT device makers yet. For instance, contemporary solutions which are trying to fix these issues we talked in the previous section (2.1) which are affecting ordinary users and their private lives, are mostly offering and suggesting users to utilize the existing features of the hardware and software tools that they are already using but not to the fullest extent. Some of those solutions or more precisely suggestions are advising users to change default admin username and passwords, use more complex Wi-Fi passwords, not use routers supplied by Internet Service Providers (ISP), apply MAC address filtering to avoid connection of unwanted devices to the network and use Virtual Local Area Networks (VLANs) inside a larger private network to isolate IoT devices which may have vulnerabilities [28]. A similar but more advanced approach has also been offered; the usage of Unified Threat Management (UTM) systems. Simply put, a UTM system is a type of network hardware appliance, virtual appliance or cloud service that protects networks from security threats by combining and integrating multiple security services such as firewall, gateway anti-virus and other intrusion detection and prevention features into a single platform [29]. UTM systems are more frequently seen in network protection of businesses rather than private home networks. However, with the proliferating domestic usage of IoT devices, a sheer number of companies, e.g. Zyxel, Sophos, etc. are now offering this high level intrusion prevention systems to even non-tech savvy individuals.

There are also some companies who adopted the UTM concept into their new generation secure routers which have a higher average market price. These have incorporated a higher-end technology and advanced features –along with the security measures mentioned above– to secure the network and the IoT devices inside it. Examples of this new generation routers are: Luma Smart Wi-Fi Router [30], Bitdefender Box [31], and Verizon [32] with the Home Network Protection software embedded. The exact same idea to protect private networks and the IoT devices inside it, is applied to smaller devices which work by being plugged into one of the empty jacks of users' home routers by an Ethernet cable. After being plugged in, these devices auto-activate themselves and start monitoring the network. Some examples of those products are; Extreme Defender Adapter [33], F-Secure Sense [34], Dojo [35], and CUJO [36]. The key features of all those smart routers and devices are the same. They monitor the incoming and outgoing traffic and block or cut the connection when an unusual or suspicious activity is detected. They also scrutinize the activities in the local area network to prevent any connected device to exfiltrate user data without permission. Thus, they come along with an administrative mobile application which could be used by the user to tune the security measures for individual devices in the network and get related notifications. Those products also adopt machine learning techniques and other advanced technologies like behavioral analysis to detect attack patterns and block new threats [37]. However, in order to achieve such a complex task a cloud service is required, to which the user data is uploaded and processed. As the article [37] indicates for Dojo that, it comes with the functionality of profiling all the network devices upon connection and sending the device activity metadata back to its cloud server for further examination. Similarly, CUJO relies on its cloud server to learn and protect against new malicious activities. It achieves this by pushing user data on its servers for again further examination in a regular basis [37]. This very feature of those products violates our most vital concern, the user data privacy. It forces users to trust and not question the fact that their private data is sent, stored and analyzed by the companies of those smart routers. How can we expect users to trust those "security" companies while they are trying to trust no one but themselves with their personal data?

Finally, another similar but well studied approach that we have analyzed worth mentioning is the usage of IoT gateways [38]. There are different applications of this idea which are usually appear in forms of either a hardware or a software [39]. In either case, the idea is to have an intermediary hardware which runs a software that fortifies the communication between the IoT device generating the sensitive data and the server that this data is being shared with. This practice is very useful when applied on the IoT systems where the IoT device generating the data, has limited computing power to utilize advanced security protocols like TLS [40]. Here the gateway is the intermediary software that receives the generated data first. Having enough computational power, this gateway establishes a secure communication with the server and shares the data over it. However, similar to the UTM adopted secure routers, IoT gateways are only helping users in establishing a secure communication between the hardware-wise weak IoT devices and the servers of the IoT device makers. So again, this technology does not help users with the problem of handing over their private data to the companies which we, users having hard time trusting in the first place.

When all is considered, those suggestions and products, even though they are helping users to secure their private networks against some type of attacks and intrusions caused by the vulnerabilities in hardware and software of the IoT devices, they are not helping users to take full control over their own personal private data. Private data is still being shared with some services where the owner of the data doesn't have any administrative rights. Thus, the issue of data privacy caused by the usage of third-party services or the illicit use of personal data by the IoT device makers remains unsolved. Though, there are some ideas promoting the importance of that. The rest of this subsection will highlight some of those proposed ideas.

2.2.1 Data as Labor

Although it is not a solution which provides full control over the data to its owners, Data as Labor (DaL) [41] is one of the most promising and unique ideas that gives some incentive such as a cash reward to users to share their personal data willingly. As it is explained in detail in Section 2.1 (*Problem Definition*), the world giants like Google, Amazon, Facebook, Apple, and Microsoft have a net profit of billions of US dollars, and increasing, thanks to targeted ads and other services specifically tailored for users. They have become so accurate about what their users want by collecting detailed personal and private information from users' activities while engaging and using their services. Since these titans' success depends on the amount of time spent by the users on their sites, they garner valuable data from these interactions. Such data is then used to improve the accuracy and quality of the targeted ads and to some degree tune their services.

The distinction between leisure and labor is becoming less and less clear. In fact, we now work all the time when tagging a friend in a picture, asking Alexa to skip the song, or even when writing a review for a place we went recently. During these activities we are actually generating the data that is required for tech companies to profit by teaching their algorithms and come up with user tailored ads and services [42]. An article in The Economist describes data as "the oil of the digital era" to emphasize its market value [43]. For this very reason, the supporters of DaL idea suggest that anyone who uses online services and provide intimate data should be paid accordingly. They also argue that, treating data as labor would not only make the society better and fairer, but also could spur the development of technology and help economy. “At present, because data suppliers are not properly rewarded for their digital contributions, they lack the incentive or freedom to contribute the high-quality data that would most empower technology or develop their personal capacities to maximize their earnings and contributions to the digital economy” says Eric et al [41].

Although the idea of DaL is rational, to date it has not been practical. For example, in 2014, an American artist, Jennifer Lyn Morone, started a company under her name Jennifer Lyn Morone™ Inc. The purpose behind that effort was to exploit her personal data for financial gain. She collected her data under different categories and in 2016 in a London gallery, she offered her personal data for sale. Having different prices for different types of her data, the total collection which includes her health data and social security number was worth approximately £7,000 [44]. However, only a few buyers were actually interested in her offer and she finds “the whole thing really absurd” [44].

2.2.2 Privacy Laws

Governments are also involved in solving issues on data privacy by enacting new laws. Those information privacy laws or data protection laws have been adopted by over 80 countries including nearly every country in Europe. On May 11, 1973, the world’s first national data protection law was enacted in Sweden under the name of *Datalagen* meaning The Data Act [45]. This act was endorsed by many other countries and led them to legislate similar laws in the following years. Parliament of Canada enacted Personal Information Protection and Electronic Documents Act (PIPEDA) on April 13, 2000 [46], and on May 25, 2018, European Union (EU) legislated the regulation called General Data Protection Regulation (GDPR) which is similarly meant for the protection of private data under EU law [47]. The purpose of those legislations is to enforce rules that protect users’ sensitive information against variety of privacy issues. Since there are different legislations adopted by countries in different continents and by independent territories, this paper will not discuss every single chapter of every one of those legislations. However, the essential idea behind all of them is to guarantee that the personal data collected from users cannot be processed or shared in any way without the owner’s consent. They also enforce the right for data owners to request the deletion of their information stored by any company. To provide a deeper understanding on the topic, a part of *Article 6* from GDPR is given below as an example:

1. *Processing shall be lawful only if and to the extent that at least one of the following applies:*
   a) *the data subject has given consent to the processing of his or her personal data for one or more specific purposes;*
   b) *processing is necessary for the performance of a contract to which the data subject is party or in order to take steps at the request of the data subject prior to entering into a contract;*
   c) *processing is necessary for compliance with a legal obligation to which the controller is subject;*
   d) *processing is necessary in order to protect the vital interests of the data subject or of another natural person;*
   e) *processing is necessary for the performance of a task carried out in the public interest or in the exercise of official authority vested in the controller;*
   f) *processing is necessary for the purposes of the legitimate interests pursued by the controller or by a third party, except where such interests are overridden by the interests or fundamental rights and freedoms of the data subject which require protection of personal data, in particular where the data subject is a child* [47].

Given those statements, the enacted regulations seem to be protecting the users and solving majority of the data privacy issues. However, when examined carefully, it can be clearly seen that almost all of the world giants like Facebook, Amazon, Google, Microsoft, Apple, and etc. who collect billions of people's data and at some point cause privacy issues are located in United States soil and incorporated under the United States laws and legislation. Although any company in the world who have European customers have to comply with GDPR, enforcing a foreign regulation and imposing penalties accordingly right away seems unlikely given the complicated nature of the rules stated in GDPR [48]. Similarly, according to the investigation conducted by the Office of the Privacy Commissioner of Canada (OPC) in March 2018, PIPEDA as well, was not enough to protect users' privacy as Facebook has failed to comply with it [49]. Moreover, as of 2019, United States is one of the several countries who have not adopted any kind of comprehensive information privacy law yet. Instead, there are certain acts and limited sectoral regulations enacted by the

United States Congress to protect users with fair information practices in some areas. Currently in 2019, some of those are; Health Insurance Portability and Accountability Act (HIPAA) [50], Fair and Accurate Credit Transactions Act (FACTA) [51], and Code of Federal Regulations (CFR) [52] which was published by the Office of the Federal Register of the United States. Essentially those acts have similar principle and enforcements with other legislations adopted by those 80 countries. However, as it is proven with a sheer number of real life examples in Section 2.1 (*Problem Definition*) that none of those tech giants seems to be complying with any of these acts or regulations to actually protect users' sensitive data. As the Apple CEO Tim Cook said during an event in California in 2018 that, even though some state laws are looking to accomplish giving users the full control over their personal data, right now there is no federal standard protecting users from such practices violating the data privacy [53].

2.2.3 Distributed Ledger Technologies

The last solution worth mentioning in *Proposed Solutions* section is the application of Distributed Ledger Technology (DLT) [54] in IoT systems. A DLT is simply a digital, decentralized and censorship-resistant system for storing transactions made by the participants of the network. These stored records are called ledgers which are kept in multiple places (nodes owned by participants), synchronized consistently, and can be accessed by any participant. By keeping multiple copies of the ledger in a distributed manner and requiring consensus of all of the participants when an update has to be done on the ledger, the system (the network) eliminates the possibility of malicious changes that can be done by a single party, and therefore, manages to guarantee the accuracy and correctness of the transactions that have ever been made. Moreover, unlike traditional databases, since any change that is to be made on the ledger requires a consensus, DLT systems require neither an administrator nor any kind of administrative function.

There are sheer number of DLT applications using different kind of algorithms adopting the distributed ledger idea. In this paper, we will mainly focus on two of these algorithms; namely Blockchain [55] and Tangle [56] which have solid applications on IoT systems. Briefly, Blockchain is a chain of blocks which holds the transactions made by the participants so far. Each block can hold one or multiple transactions (records) depending on the block size determined by the application. Every time a new transaction is made, the system generates a new block to include this new transaction and any other awaiting transactions, and link it as the last block of the chain. This chain represents the ledger and multiple copies of it are kept by the participants.

Tangle on the other hand, uses a different approach to keep a distributed ledger. Instead of a chain of record blocks, Tangle utilizes nodes positioned in a directed acyclic graph (DAG). Every node holds records of transactions similar to blocks in Blockchain. However, while the chain connects every transaction made since the beginning, edges of the DAG connects some of the nodes (transactions) to some other. Thus, Tangle and Blockchain have totally different transaction validation processes. Moreover, in order for a participant to make a transaction in Tangle, this new transaction has to validate random two latest transactions. Therefore, while Blockchain requires miners two validate new transactions, Tangle offers a miner-free system.

There are three key benefits of the DLT usage in IoT domain. It builds trust between parties and devices, reduces costs by removing any intermediary related overhead, and accelerates transactions to become almost instantaneous according to Aran, a Blockchain expert [57]. Let us elaborate. First of all, most of the IoT systems, if not all, utilize a cloud or client-server model to function and synchronize the devices that participate in the system. In such a centralized model with hundreds of interconnected devices, one of the biggest concern is the exponential increase in costs to handle the synchronization and communication of those devices. Second concern is that, since it is centralized meaning that there is only a single central authority which controls the network and protect the user data, any vulnerability in the server would jeopardize all of the relying devices. Likewise, each participant in the IoT architecture could act as a point of failure, meaning that, if an IoT device connected to a server is breached, every other device connected to that same server could easily be affected, and thus, disrupt the entire network [58]. Finally, as it is discussed in previous chapters that there is no guarantee that the collected data is put to appropriate and promised use as a cause of having a central authority in a centralized cloud or client-server model. DLT solves all those issues by decentralizing the system. As Aran says, “Since a decentralized database would remove any one point of weakness attackers would have to target individual nodes on the network instead. Any attack on an individual node on the network would also be futile as all the other nodes would resist any attempt to alter the data” [57]. To actually forge or alter the data by winning the consensus in Blockchain, a malevolent participant has to possess more than half of the whole community composing the network. However, this so-called “51% attack” is practically impossible to realize regarding the resources it requires. Additionally, since there is no intermediary involved in transactions the time required for one device to send data to another is significantly reduced. It is also a much trusted and consistent system compared to traditional centralized models since the validity of all the transactions made are verified by the participants and permanently and transparently saved on the ledger for anyone to check on a later date.

To achieve machine to machine (M2M) communication in a DLT environment, IoT devices rely on smart contracts. Ethereum [59] is one the most popular projects which adopts smart contracts. These contracts are basically decentralized applications (DApp) or procedures coded in a programming language –commonly in Solidity– and published as yet another transaction to run on a Blockchain network. Once a smart contract is published and validated, any participant in the network can invoke any public function of that application or service. Thanks to consistency and transparency properties of Blockchain, usage of smart contracts provide a lot more secure environment and model for IoT networks. Additionally, Blockchain cryptographically signs transactions and regularly verifies those signatures to ensure that every transaction is generated legitimately by the actual originator. Doing so, it is able to eliminate the possibility of man-in-the-middle, replay, and other attacks [58].

However, there are also some serious issues that come along with the advantages of Blockchain technology. One of them is the increase in transaction fees. In Blockchain technologies there are miners who earn tokens (coins) as an incentive for doing the Proof-of-Work (PoW). Part of this token comes from the fees paid by the senders and the rest comes from the "coinbase". Thus, the transaction fees increase with the diminishing number of coins held in the "coinbase" and when more computational power is required during rush hours. Paying fees for each and every transaction needed for IoT devices to communicate may be problematic, not to mention the increasing costs. There are two more big issues which make usage of Blockchain technology in IoT domain difficult. The time to process including validation and consensus of a transaction is rather slow for some IoT systems to function properly. In an experimental work of applying Blockchain technology on an IoT system using Ethereum, Seyoung et al. realized that, although the transaction time is around 12 seconds, it still was not fast enough for some domains [60].

The second problem is that the ledger size is getting bigger with every block added to the chain. Thus, majority of IoT devices may not be able to run a full node. For this reason some Blockchain platforms introduced light nodes which can do transactions like full nodes but rely on third parties –someone else's full node– for every other action needed. Since every transaction made by a light node has to be processed through a third party, this third party can access all the information about the transaction that is being processed, as well as any personal data exposed to them as a service provider [61]. Which is exactly what we try to avoid! The only solution to that is to run a full node which allocates more than 1 TB for Ethereum today. Based on their experiment Seyoung et al. says, "since light client is not supported on Ethereum at this point, either we need to use a proxy or have a large storage to save entire blockchain. Using proxy may be easy. But we compromise security because there is a third party involved. Second solution, while it does not compromise security, may require large storage, which would be too expensive or infeasible for small IoT devices" [60], to support our argument.

Tangle is introduced as another type of DLT to address these problems stated for Blockchain technology. IOTA is one of the Permissionless Distributed Ledger Systems using Tangle technology. This trending platform argues in their webpage that the Blockchain networks become more and more centralized around some powerful actors with the increase in competitiveness of getting financial rewards for validating transactions. However, they say, the need for decentralized and permissionless systems remains and is increasing each year [62]. Compared to Blockchain technology, Tangle is built as a Directed Acyclic Graph (DAG) and every node (participant) in IOTA contributes to the consensus of the network by approving random two past transactions (a tiny PoW) to be able to do a new transaction. Therefore, IOTA nodes are more lightweight since they only need a portion of the total of the past transactions to validate the latest two and since no miners are needed, there are no transaction fees as well. Also, since every new transaction has to do a tiny PoW in order to be added to the network, the higher the load is the faster the IOTA network gets, and therefore, technically can scale infinitely and handle unlimited amount of transactions per second. However, as much as it sounds perfect to be the backbone of the IoT systems, IOTA or Tangle technology has its own problems too.

According to an article published by Medium says that the Tangle by itself cannot provide an effective solution against so-called "34% attack" (double spends) at low loads [63]. IOTA also utilizes a mechanism called Coordinator controlled by the IOTA Foundation to prevent double spends, however, this gives them a certain amount of control over the network and thus centralizes the whole system [63] [64]. Another claim of IOTA is to be quantum immune by utilizing a trinary hash function called Curl. However according to the same previous article, usage of such a complex function increases the size of a transaction to almost ten times and when it comes to scalability, it seems counterproductive [63]. Finally, smart contracts is the essential part of DLT to be adopted by the IoT systems. However, DAG ledgers like Tangle are not smart contract friendly because of their innate structure [65]. Although, other platforms like Vite [66] or Qubic for IOTA [67] are trying to achieve a similar feature, all those technologies are still under development and there is no guarantee that there will not be any hidden vulnerabilities as a result of their complex nature.

When all is considered, despite of all the issues with the technologies akin to Tangle or Blockchain, the DLT is a revolutionary idea and a candidate to be the backbone of the IoT domain. However, no matter how it is implemented, DLT's core logic and running principle is always the same. For example, there will always be a need for consensus to add a new transaction to the network or there will not be any kind of central authority or data in the ledger can never be altered or deleted. Even though these features are what makes the Distributed Ledgers trustable and secure than any other system when it comes to personal data, they may also be the ones which may cause for any DLT platform to be unusable in real life applications. For instance, when an information is stored in the ledger it becomes permanent, which means that if an action is taken by mistake through a smart contract, there will not be any authority where one can report this unwanted action to be reverted. Another unignorable problem with DLT is the introduction of GDPR which we presented in *Privacy Laws* subsection. GDPR's *Article 16, Article 17,* and *Article 18* are clearly conflicting with those properties of DLT. To begin with, *Article 4,* the *Definitions* section defines personal data as follows:

1) *'personal data' means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person;* [47]

Here the statement is a little bit broad and does not indicate or cover any specific kind of information which can be used to identify a person directly or indirectly. So, while name, age or gender of a person can directly identify this individual, phone number, credit card number, IP address or even a DLT platform's account number (wallet address) can indirectly do that. That means, any DLT application with EU citizen users lies within the scope of GDPR and has to comply with it. Given this fact, let us continue with *Article 16, Right to rectification.* This article gives data owners the right to change an inaccurate personal data or complete any incomplete personal data by means of providing a supplementary statement [47]. However, while one can add a new data to a distributed ledger, not being able to change or update any stored data is a big problem. The same applies to *Article 17, Right to erasure ('right to be forgotten').* It simply says that under the right circumstances, data owners have the right to ask for deletion of their personal data [47]. Since a stored data also cannot be deleted from a distributed ledger, a DLT user cannot exercises his/her right to be forgotten. Therefore, in order for a DLT platform to comply with GDPR, it must not store any personal data of a EU citizen. Finally, *Article 18, Right to restriction of processing* says that under the right circumstances including the unlawful process or inaccuracy of the data, owner of that personal data can restrict the usage and processing of it [47]. However, most of the DLT platforms are transparent meaning that any participant can get a copy of the ledger and do anything with the data stored in it without the owner's permission or even knowledge. Solutions like encryption of the data, utilizing proxy servers to store the actual data or using DLTs akin to Permissioned Blockchain[8] either centralize the network or do not comply with some GDPR articles. Currently, the only work around on those issues while complying with GDPR is the usage of Zero Knowledge Proof (ZKP). ZKP lets users to prove that they possess the information in question to another source without revealing any part of that information to the source [68]. Unfortunately, not even ZKP is flawless. Although it has only one drawback, it is a serious one: Only the owner of the information (e.g. password) has a copy of it. Meaning that if the owner loses or forgets the information it is unrecoverable, irretrievable, and thus, gone forever [68].

[8] Permissioned Blockchains maintain an access control layer to allow certain actions to be performed only by certain identifiable participants [87].

2.3 Conclusion

Overall, these solutions that are proposed so far have some potential and are promising to some extent. However, at the time of this project they either still have some undeniable issues that need to be resolved in order to be practical, and thus, realizable, or they simply are not addressing the root cause of the data piracy problem which is sharing the private user data with IoT device makers in the first place, and therefore, losing the control over it. Moreover, although we do think that DLT is a fascinating technology, we also consider it as an overkill for small scale IoT systems beside of its problems in IoT domain. Regarding their power consumption along with their computational and communicational requirements for almost every action (transaction) taken, DLTs become an unnecessarily heavy tool for managing a limited set of personal IoT devices which should not be exposed to people other than their owners in spite of their internet connection. Therefore, if we are to keep our scope on small scale home usage for a number of personal IoT devices, a highly secure communication between the users and their IoT devices by means of encryption seems both an efficient and a sufficient solution. Hence, considering this, we are proposing a lighter, robust, and system-based solution called Heimdallr in the light of some of the ideas discussed above, which would not let the data leave the user’s private network unlike the DLT systems.

3 Heimdallr System

Along with the contemporary problems that exist in IoT networks, there would be other serious issues with their proliferation and usage in people's private lives e.g., IP cameras, baby monitors, smart menstrual period trackers etc. For example, even though the baby technology sector is relatively new and unstudied, the market research firm Hexa is estimating the growth in baby-monitoring sub-market alone to be from 929 million US dollars in 2016 to 1.63 billion US dollars by 2025 [69]. Therefore, from what has happened in other areas where user data is up for grabs, with this dramatic increase, it is likely that the data produced by the IoT devices adopted in these sensitive fields could also be hijacked and may lead to worse consequences. According to a survey conducted by Nicholas Shields in 2018, 87% of the respondents already feel uncomfortable using services like Amazon Key which utilizes IoT devices like smart locks and 88% feel the same with the usage of their personal data to deduce other sensitive information about them [70]. However, there is no doubt that the technology makes our lives easier even though they compromise our privacy. At this point people either have to accept the fact that they don't have any control over their personal data collected when using IoT products or they have to stop using them completely. Instead, a third option must immediately be introduced which provides a solution to data piracy without giving up on the technology that ease our lives.

To address the problem of IoT data piracy, we propose a system we have named Heimdallr to create a fortress around users' private networks to protect the personal data generated and used by IoT devices, and allows the users to take control over it.

3.1 Heimdallr

Heimdallr is a software system that would run on a low cost processor in the users' next generation personal wireless smart modem/router: itself an IoT. The reason why we sought a solution in the modem/router is, this is required to connect most of the IoT devices to the Internet and hence it is the head of the snake; the bridge connecting the IoT devices to the digital jungle, the Internet. This is supported by a study concerning IoT security and privacy conducted by Nick et al.; it reports, "Preliminary results indicate that home gateway routers or other network middle-boxes could automatically detect local IoT device sources of DDoS attacks with high accuracy using low-cost machine learning algorithms" [71]. This also means that any flaw hidden in any of the IoT devices in the network that can threaten both the user privacy and the other connected IoT devices can be avoided by preventing any malicious data to infiltrate or any personal sensitive data to exfiltrate at the modem/router.

The main idea behind the Heimdallr system is to allow users to take full control over their data by, first, monitoring and manipulating the network traffic on the modem/router and secondly, running all the necessary services locally (in the modem/router) for users to have access to full functionality of their connected IoT devices without the need of establishing a connection to any server of any of the IoT makers. This feature resembles the smart contracts of Blockchain. The only difference is that a smart contract is stored on the Blockchain network where any participant can access and use. On the other hand, Heimdallr has its own memory space to install and run any service that is required for connected IoT devices to function properly, and is normally provided as a cloud service on IoT device makers' servers. Additionally, Heimdallr users have their own private storage to store their personal IoT related and/or generated data. Hence, it can be seen as a private server providing cloud services for the owner's connected IoT devices. The diagrams provided below (Figure 1 and Figure 2) clarifies this concept with a basic comparison:

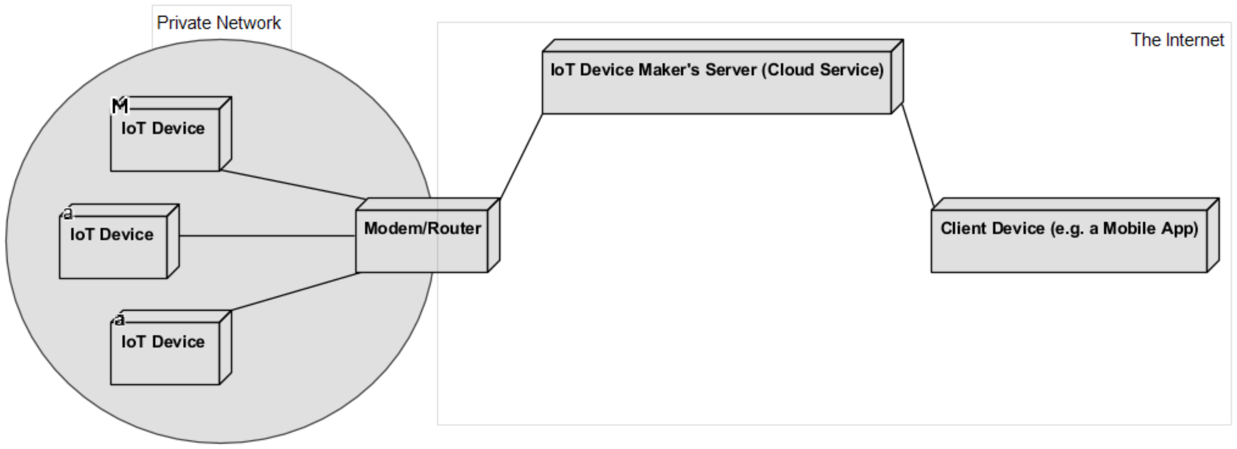

Private Network
The Internet
IoT Device Maker's Server (Cloud Service)
IoT Device
IoT Device
IoT Device
Modem/Router
Client Device (e.g. a Mobile App)

Figure 1: Traditional Concept of IoT Systems
This figure shows the traditional way of creating an IoT system. The IoT devices in the private network are connected to the modem/router which is the gateway to the Internet. The communication between these IoT devices and their user is provided by the IoT device maker's server and the cloud service runs on it.

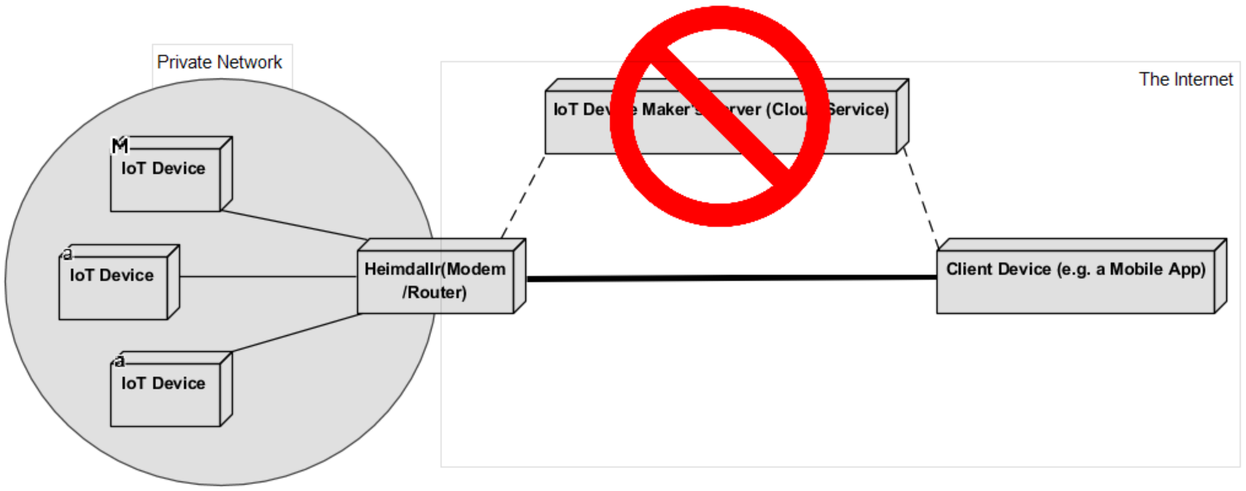


Figure 2: Heimdallr Concept of IoT Systems
This figure shows the Heimdallr version of creating the same IoT system given in Figure 1. Here we see that the IoT device maker has no longer got a role in establishing the communication between the IoT devices and their user. Instead, Heimdallr software that is running on the private network's modem/router provides a direct connection between them.

This solution would require IoT device makers to develop Heimdallr compatible user interfaces and services along with their IoT products. IoT device makers are currently obliged to provide software for one of the current tech monopolies, however, they would have to adapt to the users' demand for secure IoT devices.

Since Heimdallr is the gatekeeper, all interactions with any IoT device are under its guard, including those coming from mobile phones! This obliges us to state the underlying requirements for the system.

3.2 Assumptions and Requirements

Since what Heimdallr is trying to achieve is somewhat complex, it has three main requirements to be fully functional. Those requirements are; introduction of a *Software Assurance Agency,* utilizing *new generation modem/routers,* and the *acceptance and support of the concept.*

3.2.1 Software Assurance Agency (SAA)

This is the central quality assurance agency and would be an arm's length non-profit software certification organization which we denote as "Software Assurance Agency" or "SAA". It is introduced to be responsible for testing, inspecting, storing and certifying software ranging from device interfaces and updates including those provided for IoT devices by their device makers. The objective of this organization is to ensure that all software have adequate privacy measures and would not threaten users' private data. It is worth pointing out that currently in the software domain, all this is done by the organization which manufactures and markets the IoT; these corporations currently have no public oversight. The only beacon is the open source community where dedicated developers donate their time, talent and energy to produce open source software which is often free and the source code is accessible to anyone to investigate and verify their algorithms and functions.

SAA also monitors all software updates and prohibits any that alters the operation of any connected IoT device by introducing unwanted features without users' consent. The consent requires a rational that is easy to understand. Thus, SAA will guarantee that all software certified by it, does exactly what it is supposed to and does not pose any threat to users' privacy. Put simply, SAA introduces some measure of standardization for IoT devices as have been done by organizations such as CSA [1] and UL [2]. We envisage that SAA would remain independent and not become a front for IoT industry as has been the case with some of the recent organizations which are directly or indirectly, essentially public relations surrogate for the industry [72] [73]. And unlike any other software stores currently being used by consumers to download applications and their updates, SAA would not be an extension of the tech giants who run their own software stores for profit while assumed to be maintaining some quality.

3.2.2 New Generation Modem/Routers

The second requirement is to have static IP assigned to our 'smart' modem/routers equipped with higher-end hardware compared to the current routers and it would include a computing system[9]. This is required, since, Heimdallr is intended to be run on users' modem/router which is the gateway for IoT devices to the Internet; it must have the minimum processing capability to run the Heimdallr software and the required services of the IoT devices as discussed earlier in this chapter. With a high-end hardware the necessary computational power can be provided to handle complex tasks and most importantly multiple tasks given the fact that not only one but dozens of IoT devices will be using Heimdallr services, therefore its resources. And since the communication between the users and their IoT devices will be established and monitored by Heimdallr, users should be

[9] We envisage Heimdallr to be equipped with an economic computing device such as Raspberry Pi and its own storage. What we are aiming for is the pendulum to swing back and bring the user data back home from the cloud!

able to connect –remotely– to their Heimdallr. Hence, it has to be assigned a static IP address to achieve this. Without a static IP address users won't be able to know the current IP address assigned to their Heimdallr. Even though there are some work arounds to achieve this while having a dynamic IP address, that would be an unnecessarily complex and an inefficient task to get the same result, moreover, it may even compromise user privacy. More on the hardware requirements will be discussed later in Section 5.2 (*Analysis and Assessment*).

3.2.3 Acceptance and Support of the Concept

The final issue that this system has to deal with is the need to be accepted by the global industry. Under the current circumstances, most of the companies, especially the technology giants, would not want the supervision of Heimdallr and SAA, given the profit that they are reaping from users' private data. Regarding that, imposing either administrative or social sanctions or both can be considered as a possible solution as long as there exist users who are concerned about the privacy of their data. In any case, such concerns would create legal, societal and competitive pressure for IoT device makers to support and use a certification authority such as SAA.

3.3 Issues Addressed

In this section, the problems that the Heimdallr system addresses are explained in detail. We divided those problems threatening the data privacy into two subsections; *IoT Device Maker Related Issues* and *Hardware/Software Related Issues*.

3.3.1 IoT Device Maker Related Issues

As discussed in Section 2.1 (*Problem Definition*), recent experiences with manipulating user data for political gain amply illustrates that unregulated tech companies and the third parties they are associated with, are not necessarily the organizations that should be trusted with privacy and security of users' data [74] [75]. As Apple CEO Tim Cook said in 2018, during the same California event mentioned earlier, "One of the biggest challenges in protecting privacy is that many of the violations are invisible. For example, you might have bought a product from an online retailer—something most of us have done. But what the retailer doesn't tell you is that it then turned around and sold or transferred information about your purchase to a "data broker"—a company that exists purely to collect your information, package it and sell it to yet another buyer" [53]. Although not every company is selling user data for profit, nonetheless many of them still have other problems.

Third-party services are one of the most common examples of this issue. Many small and big IoT companies use third-party services in their products to provide additional features to users without their consent. The same study mentioned earlier on IoT devices shows along with many other examples that within the first minute after turning on the Samsung Smart TV, it talks to Google Play, Double Click, Netflix, FandangoNOW, Spotify, CBS, MSNBC, NFL, Deezer, and Facebook—even though the user did not sign in or create accounts with any of them nor may want to [71]. It also indicates that the Geeni Smart Bulb communicates with gw.tuyaus.com which is operated by a company called TuYa. This company also provides an MQTT service, which eventually allows the manufacturer to communicate with their IoT device in users' home without the users' knowledge or permission [71]. Therefore, not only users have lost the control over their data, there is also no guarantee that these third-party services will not do anything harmful with it. For all these reasons, Heimdallr system requires all IoT device makers to develop Heimdallr compatible user interface (UI) and services that will run locally. This would produce two desirable results. First, none of the IoT devices will be required to use any known or unknown services on the Internet, which means that user data will flow only between the user and the IoT device via Heimdallr and go nowhere else. Second, all of the data that the IoT devices consume can be monitored, changed, collected and deleted by the user with the intermediary Heimdallr. As a conclusion, there will be no reason to have unwanted data leaks since users will have the full control over their data.

Another problem with keeping hundreds of thousands of user profiles in company databases is that it basically creates a bigger risk of data theft. People do not just use one service or a single IoT device, but they use products of dozens of different companies. Even with a single product, a user's data gets stored in the device maker's database, as well as databases of all the third-party services that this product works with. Therefore, in a real life case where a single user owns several products, duplication and storage of their private data in a number of locations increases gradually. This raises the threat to data privacy, since, the same user's data is stored in an unknown number of different databases with different security measures and vulnerabilities. While, one company might be able protect users' data in a difficult situation, another one could fail to do so under the exact same circumstances. Similarly, an article posted on Electronic Privacy Information Center's website says, "In another sense, control can be lost as more and more companies collect data about users. This data often paints a detailed picture of individual users through the collection of activities online" [76]. This indicates that keeping the personal data in one place is definitely a step that must be taken on the way to secure it. With Heimdallr user data will only be stored locally in Heimdallr's database. Thus, one can be sure that the data is secure as long as the Heimdallr is. Additionally, since every user will have an independent Heimdallr system, there will be less data stored in any database. As Apple CEO Tim Cook says, it is not a good idea to keep all in one place, since, it would be very bad for security and privacy [77]. Hence, when the effort needed to penetrate billions of Heimdallrs is compared to its reward, it becomes less attractive for intruders who are seeking to violate others privacy.

3.3.2 Hardware/Software Related Issues

IoT related problems do not always stem from companies' (IoT device makers') privacy policies. They sometimes arise from rushed or imperfect software development which lacks essential security measures. Non-technical or even the technical users may not be able to notice a defect in software which leads to serious violation of data privacy. Given that software is embedded and running in IoT devices, it becomes nearly impossible for most users to analyze them. Additionally, most of the IoT devices do not even mention the specific third parties they communicate with in their privacy policies. This makes it nearly impossible for the consumers who are seeking to make a healthy purchasing decision based on security and privacy considerations [71]. Introduction of SAA plays an essential role here; trustfulness of the software used in IoT devices is guaranteed for maximum safety of personal data and privacy by a not-for-profit open organization. Hence, it can also be seen as an ultimate guide protecting the potential IoT users by informing them about the infrastructure of the products which they are planning to buy.

Every IoT device consists of both software and hardware, and therefore, both aspects have to be considered for probable vulnerabilities. An article from Malmö University that supports this idea says, the appliance manufacturers are not always as rigorous as established software developers when it comes to security and quality criteria during the development phase. On the contrary, they develop the embedded systems in IoT devices using existing chips and designs, therefore, the security and quality criteria in their development process is usually unknown [78]. For that matter, while SAA is offering a solution to software defects, Heimdallr will be protecting the integrity of user data against any kind of hardware exploits to prevent unwanted breaches. Heimdallr achieves this by monitoring and controlling the incoming and outgoing data traffic to and from the IoT devices in the network. It will allow only the requests from IP addresses which are either whitelisted or not blacklisted with valid cookies, while blocking all the other data flow to and from any device. These cookies are simply randomly generated integer values assigned to users per client device (usually a web browser) that they use to access their Heimdallr remotely. Every time they attempt to login, the request packet includes that cookie stored in this client device to be verified by Heimdallr for extra security. If there isn't any cookie saved on the device that means the user is using this client device for the first time to access his/her Heimdallr. Then, a cookie must be generated and sent to the client device to be stored for later use. How we create these cookies in such a –what we call "first time login"– situation will be discussed later. Heimdallr also provides its users a secure access to IoT devices connected to it by building a firewall which checks user credentials. Thus, even if there are any hardware issues that pose a risk to user privacy, without the permission of the actual user, the integrity of private data will be ensured.

Enforcing generalized security measures can be pretty difficult when it comes to IoT devices. One of the techniques used is to create a generalized protection mechanism and if it works on the test system, it is assumed to work on all similar systems. It is assumed that this principle would work for IoT devices as well; however, since IoT devices borrow heavily from existing technologies and have to be used in differing environments the adaption is not straight forward. Thought about security and privacy are not paramount in the design and implementation of the core components, of both hardware and software of IoT devices [78]. Thus, instead of per device protection, Heimdallr offers a generalized solution by creating a safe zone for all IoT devices in its network.

A final software related issue worth mentioning is the complex nature of the security procedures involved in the server connections established between the user, IoT device and the server that provides the necessary services. Instead of depending on unknown procedures, and protocol for security and privacy of each type of IoT devices, it is preferable to come up with an open standard and its implementation to avoid denial of service or privacy breaches [79]. Thus, Heimdallr provides the necessary protocols and services to its established users securely. That means, instead of thousands of clients trying to access the same service on a server for a given IoT device, only a limited number of authenticated users would have regular access to the IoT devices. Direction of the authenticated users' request into their dedicated Heimdallr system by-passes the IoT device manufacturers' server. The only load on their server would be the requests to the SAA for registering updates to firmware and creating certificate for these updates. Heimdallr would then make periodic requests for updates to the SAA.

This solution that Heimdallr provides solves another serious contemporary issue with IoT devices, namely Distributed Denial of Service (DDoS) attacks. In August 2016, a botnet called Mirai was first identified by the whitehat security group MalwareMustDie. Right after, in September 2016, according to Kolias et al [80]. Website of Brian Krebs, a computer security consultant, was hit with 620 Gbps of traffic. The principle behind Mirai botnet is to first scan random public IP addresses through the telnet protocol (TCP port 23 or 2323) and try to log in to any IoT device discovered using just 62 possible default username-password pairs. Once the access is gained, Mirai gets control of that device to be used for further DDoS attacks. However, Heimdallr natively prevents this thanks to its firewall. In order for a request to reach an IoT device in the Heimdallr network, the requester has to first successfully enter the user-defined credentials, and secondly, the origin of the request must have a valid cookie. If a malware like Mirai botnet try to intrude on an IoT device connected to Heimdallr, it would cut the connection immediately with that origin and blacklist it. Furthermore, even if an IoT device gets infected by such a malware, a DDoS attack cannot be launched with that device, since, Heimdallr would not allow any data to pass through unless the admin allows that specific IoT device to communicate with the unknown target IP address by whitelisting it.

3.4 Conclusion

To sum up, so far we have seen how users easily lose control over their sensitive and private data in the jungle of variety of problems that comes with IoT devices, sometimes without even noticing. Although, it has three unignorable requirements, should they are met –which we believe none of them are impossible to achieve–, with the help of SAA, Heimdallr can assure an unmet, definite and robust solution unlike other proposed solutions we have discussed in Chapter 2. In the next chapter, we will be presenting system decisions and the architecture of the Heimdallr system in detail in order to give an insight into the solution that we are proposing.

4 Architecture of Heimdallr System

In this chapter, we will provide high and low level designs of the Heimdallr system which was used for our proof of concept. In order to make the full system comprehensible with ease, we will first start by explaining our design decisions using UML diagrams. After drawing the big picture, the tools and the technologies used to develop Heimdallr prototype during the process will be explained. And in the following chapter, an intuitive flow of Heimdallr system is going to be presented as a visual to fully clarify the idea. Furthermore, we will analyze the system to see what the specifications (hardware specifications, bandwidth requirements etc.) of our smart router has to be to leverage such complex tasks and assess if it is actually feasible to produce the real product and use it on our private networks.

4.1 System Diagrams

In order to see how the final product may look and function, we have implemented the essential components of a real life scenario of an IoT device usage supported by the Heimdallr system. In its simplest form, the initial look at the system is given below in Figure 3:

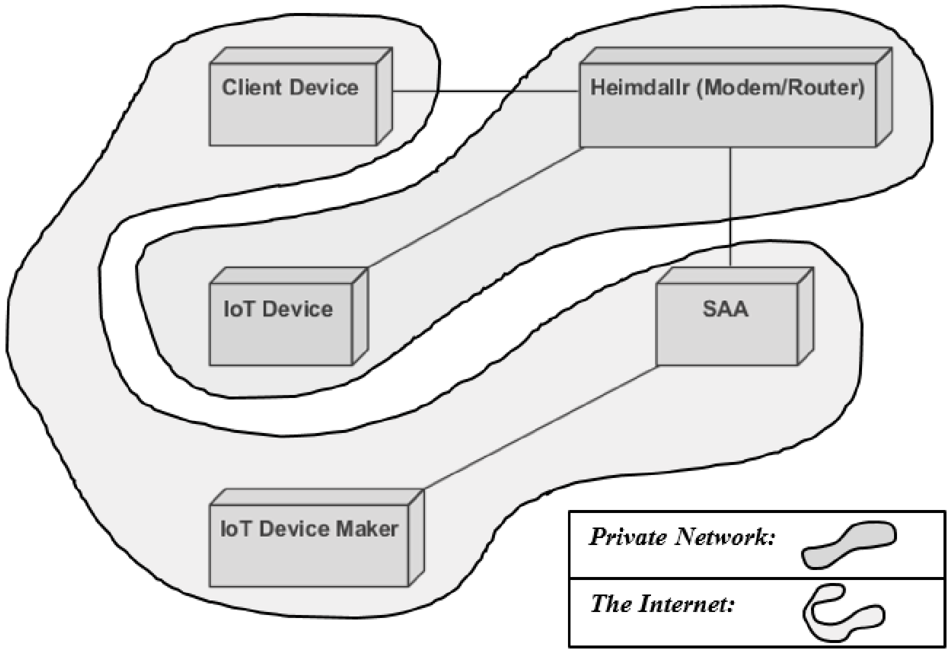


Figure 3: Conceptual Diagram of the Heimdallr System
This figure shows the relationship between the major actors of the complete Heimdallr system

To articulate this architecture, first some high-level and then low-level UML diagrams will be presented. Figure 4 gives the deployment diagram of the Heimdallr system, Figure 5 gives the use case diagram, Figure 6 gives the activity diagram, Figure 7 gives the sequence diagram, figures Figure 8 to Figure 12 depict class diagrams and finally Figure 13 and Figure 14 show ER diagrams of our proof-of-concept system. Note that these diagrams represent not an actual implementation of the proposed solution but a proof-of-concept version of the components which have a role in the design. This includes the implementation of generic *Client Device, IoT Device, Heimdallr, IoT Device Maker,* and *SAA*.

In this chapter the words *IoT User* and *Client Device* are used in close relation. To be specific, *IoT User* is a user of some IoT devices protected under Heimdallr's guard, while, *Client Device* is a client-side program which allows this user to communicate with his/her IoT devices through Heimdallr. This client-side program may be and usually is a web browser along with the key generator program that we have created. It is also important to mention that the *IoT Device* code acts as a library for any kind of IoT device we would want to implement and include in our proof-of-concept simulation. It has all the basic functionality to communicate with *Heimdallr* with an abstract task handler class adaptable according to the actual purpose of the device. For example, for our test cases we simulated a smart thermostat and a smart television using the *IoT Device* code; these would be tested as Heimdallr protected IoT devices. Note that only *IoT Device* code is shown in this paper, since all of the simulated IoT devices use the same code (library). *IoT Device Maker* is introduced to model a real IoT device maker which publishes updates and software for their IoT products. Finally, *SAA* models our not-for-profit organization, SAA. Every software that *IoT Device Maker* publishes is sent to *SAA* for inspection and certification, and once the process is successful, *Heimdallr* can fetch and install it to be used by the corresponding IoT device connected to it. Although, in our proof-of-concept version *Heimdallr* already comes with all of the software that can be published by *IoT Device Maker*, we simulated this update functionality of *Heimdallr* by checking and updating the version numbers of both the interface installed in Heimdallr and the software installed in the IoT device every time *SAA* is queried and a new update is received in return.

4.1.1 Deployment Diagram

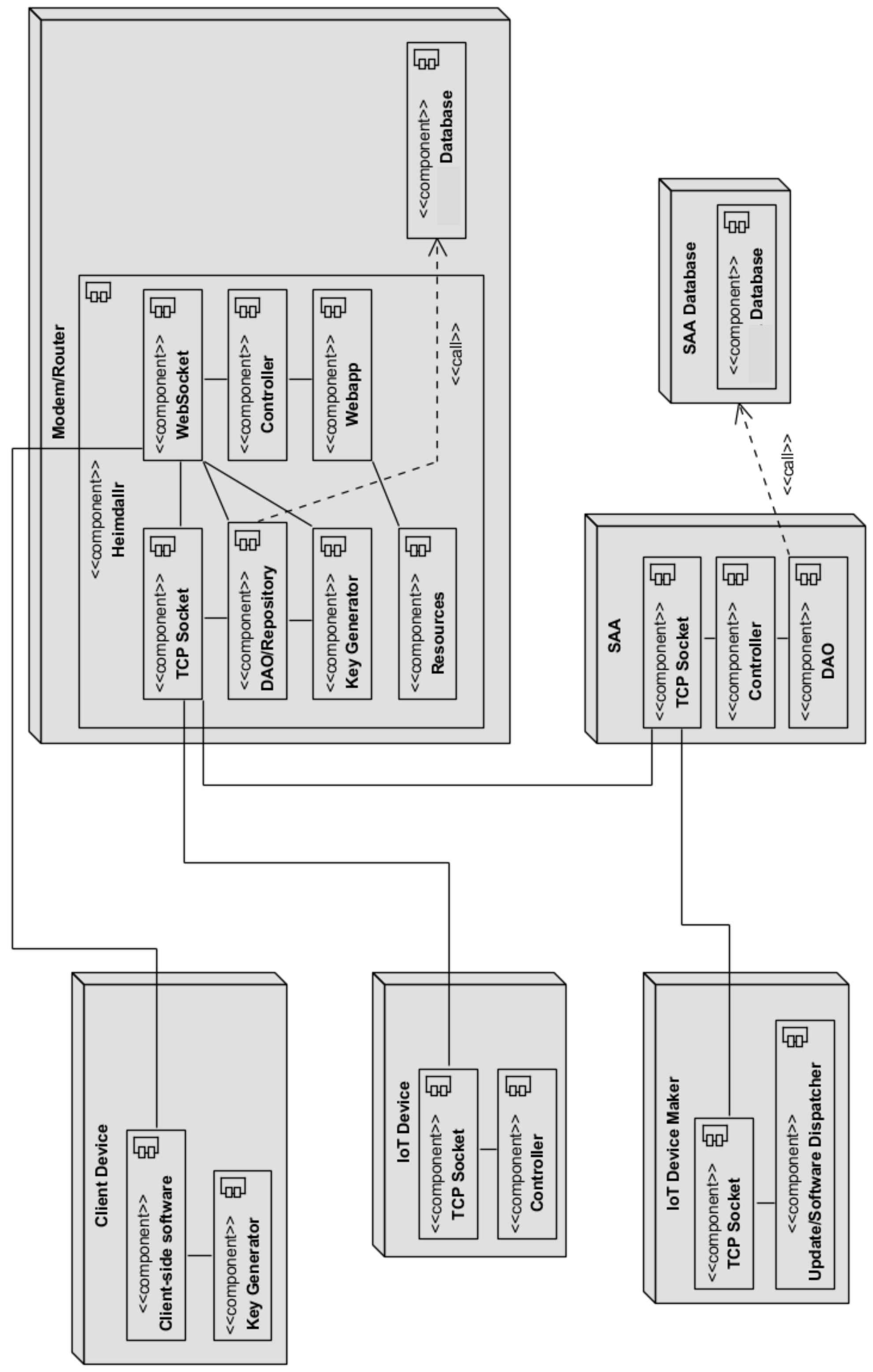


Figure 4**: Deployment Diagram of the Heimdallr System**
This figure shows the relationship between hardware and software components of the complete Heimdallr system

In the deployment diagram given in Figure 4, the overall system design and the relationship between all of the involved hardware and software components are indicated. The system is composed of mainly five hardware components plus a database for *SAA*. Those are *Client Device*, *IoT Device*, *IoT Device Maker*, user's home smart *Router*, and finally *SAA*. A client device is what enables users' to connect to their Heimdallr installed router. It hosts the key generator software that generates access codes for the first time logins on new devices and another client-side software such as a web browser or an application with a UI to interact with their Heimdallr. An IoT device is an internet connected device like a smart thermostat which is to be controlled by the client device in our tests via Heimdallr. It consists of a single program which allows the device to connect to *Heimdallr* through its *TCP Socket* and processes every task –in its *Controller*– sent by the *Client Device* to *Heimdallr* and then redirected to it. Thus, no IoT device can connect directly to the Internet.

IoT device makers, instead of accessing their IoT devices directly would simply change their procedures; the change requires them to register their products with the SAA and upload all initialization and update codes to the SAA along with all necessary documentation changes implemented and the reasons for the changes. The documentation must be public, open and accessible from the device maker or SAA. SAA is the main source for all software and their certification; all software for all SAA registered IoT devices is accessible from it.

Once any software from *IoT Device Maker* initiates a certification process with the *SAA* and if the certification process is successful, the software is ready for distribution; if not the *IoT Device Maker* is notified and the software is in a hold status until a satisfactory update is made for, and a new request for certification. A certified software is passed on to the *DAO* (*Data Access Object* – an object that is responsible for database queries, meaning that it carries data between the database and the application itself) of the *SAA* and required details are stored in the *SAA's* database. *SAA* is also connected to *Heimdallr* through its *TCP Socket* and responsible for delivering the requested software (product) or update to *Heimdallr* when needed.

The final component shown in Figure 4 is the Heimdallr router. Here the router is the high-end hardware running the *Heimdallr* software and a database connected to that. There are seven main components that we can classify in *Heimdallr*. *TCP Socket, WebSocket, DAO/Repository, Controller, Key Generator, Webapp,* and *Resources. TCP Socket* allows it to connect to the other elements, particularly to *IoT Device*s and *SAA*. *WebSocket* (WS) along with *Webapp* component provide a real-time UI –stored in *Resources*– to the *Client Device*. As before, *DAO* is responsible for database related operations such as checking the user credentials or adding a new IP address to the white-list. *Controller* is the heart of the *Heimdallr* where every request made to it is processed. It redirects users to correct pages, manages all the security related tasks, and passes all required information to and from the *Client Device* and the *IoT Device*. The *Key Generator* is the counterpart of the *Client Device Key Generator,* more about this will be discussed further in *Class Diagrams* Subsection 4.1.5.

Now that we took a first high-level look at the overall system design, before going into lower-level design decisions and their explanation, we would like to continue with a couple of other high-level UML diagrams to thoroughly deliver the idea of how the system runs. Use case, activity and sequence diagrams (respectively) will help us achieve that.

4.1.2 Use Case Diagram

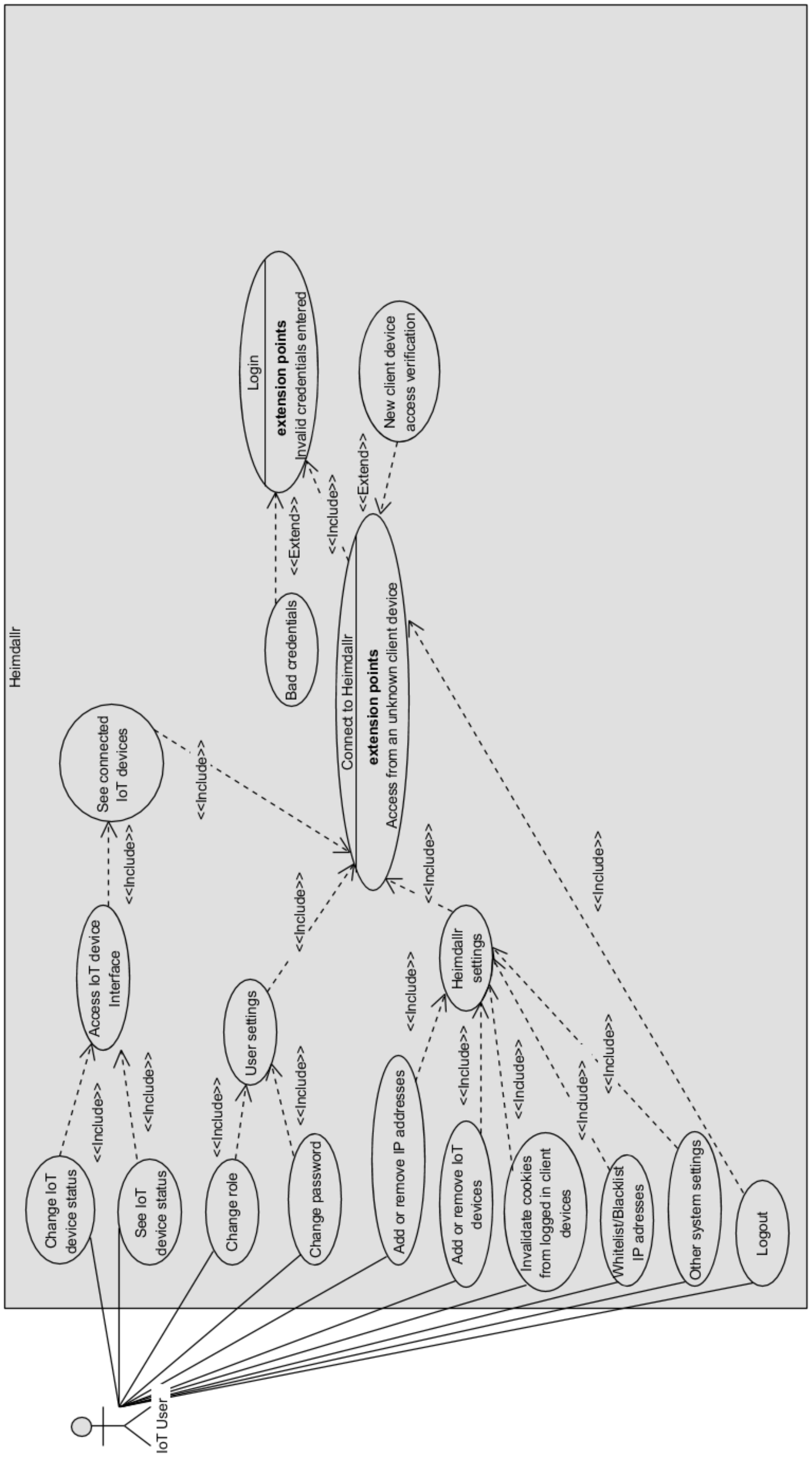


Figure 5: Use Case Diagram of the Heimdallr System
This figure shows the interaction of an IoT user with the Heimdallr system

In the use case diagram given in Figure 5, the interaction of a user with Heimdallr and the possible actions that can be taken by the user using a client device are shown. A user can control an IoT device by means of inspecting and altering its current status. This is done through the IoT device's interface. Once the user accesses that specific device's interface installed on Heimdallr, the provided UI guides the user for any possible action that can be taken which ultimately creates and sends a task to be handled by the receiving IoT device. We call any message sent from Heimdallr to any connected IoT device a task. For instance, a task can be a message sent by Heimdallr to a smart thermostat which carries a user-set room temperature data or a status message sent by the smart thermostat which tells Heimdallr the current temperature of the room. As a reminder, different IoT devices have different and specially designed interfaces that can be accessed through Heimdallr. Those interfaces are certified and provided by SAA, and downloaded and installed to become a part of the Heimdallr software.

Users can see all of the owned connected IoT devices on the main page displayed on the client device. "Owned IoT devices" means that not every user of a given Heimdallr is able to see and access every connected device. There are currently two types of user roles in Heimdallr; *USER* and *ADMIN*. Depending on the user's role, access to someone else's device can be prohibited, or moreover, not shown at all. Users however, can request and have their currently assigned role changed by an *ADMIN* user. They can also change their username and password, add or remove IP addresses and whitelist or blacklist them to restrict any unwanted data exchange between Heimdallr and any other online source, see and sign out from client devices previously used to access Heimdallr (e.g. a web browser) by invalidating cookies, connect new or remove registered IoT devices, login and finally logout. Also note that in the diagram above (Figure 5), it is shown that every action requires the user to first connect to the Heimdallr and then login before taking any other actions. Login action can lead to another action called bad credentials. This can happen when the user enters a wrong username-password pair or tries to login for the first time on a client device. In this case, the user is prompted to either re-enter the correct username-password pair or enter an access code generated with the key generator installed on the client device. Once the access is granted user is redirected to the main page where the list of connected IoT devices are shown. From this page any owned IoT device's interface can be accessed for further interaction as explained so far.

4.1.3 Activity Diagram

In this subsection we will take the use case diagram given above one step further and explain the sequence of some actions taken by the user and reactions of the rest of the system components. Note that in Figure 6, only a scenario where the *IoT User* is the leading actor is drawn. Hence, some of the preliminary actions taken by *Heimdallr*, *IoT Device*, and *SAA* to realize the *IoT User*'s actions are not shown given the fact that they are not directly related to the *IoT User*. As a result of that, an actor for an IoT device maker is not included at all, since it has no relating role. However, the same scenario given in Figure 6 will be followed in the next subsection (*Sequence Diagram*) and cover those preliminary actions as well.

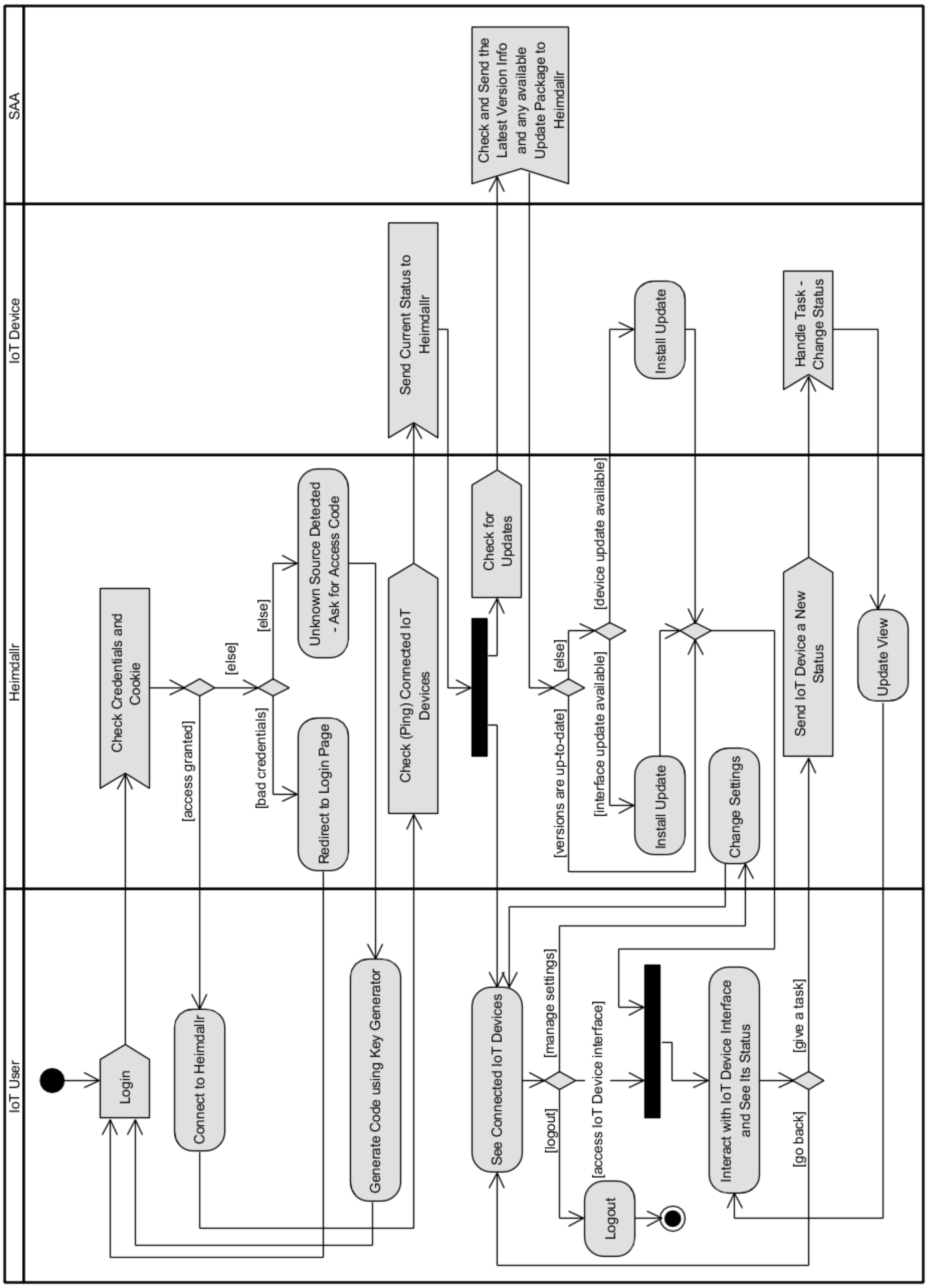

IoT User
Heimdallr
IoT Device
SAA
Login
Check Credentials and Cookie
[access granted]
[else]
[bad credentials]
[else]
Redirect to Login Page
Unknown Source Detected - Ask for Access Code
Connect to Heimdallr
Generate Code using Key Generator
Check (Ping) Connected IoT Devices
Send Current Status to Heimdallr
Check for Updates
Check and Send the Latest Version Info and any available Update Package to Heimdallr
[versions are up-to-date]
[else]
[interface update available]
[device update available]
Install Update
Install Update
Change Settings
See Connected IoT Devices
[logout]
[manage settings]
[access IoT Device interface]
Logout
Interact with IoT Device Interface and See Its Status
[go back]
[give a task]
Send IoT Device a New Status
Handle Task - Change Status
Update View

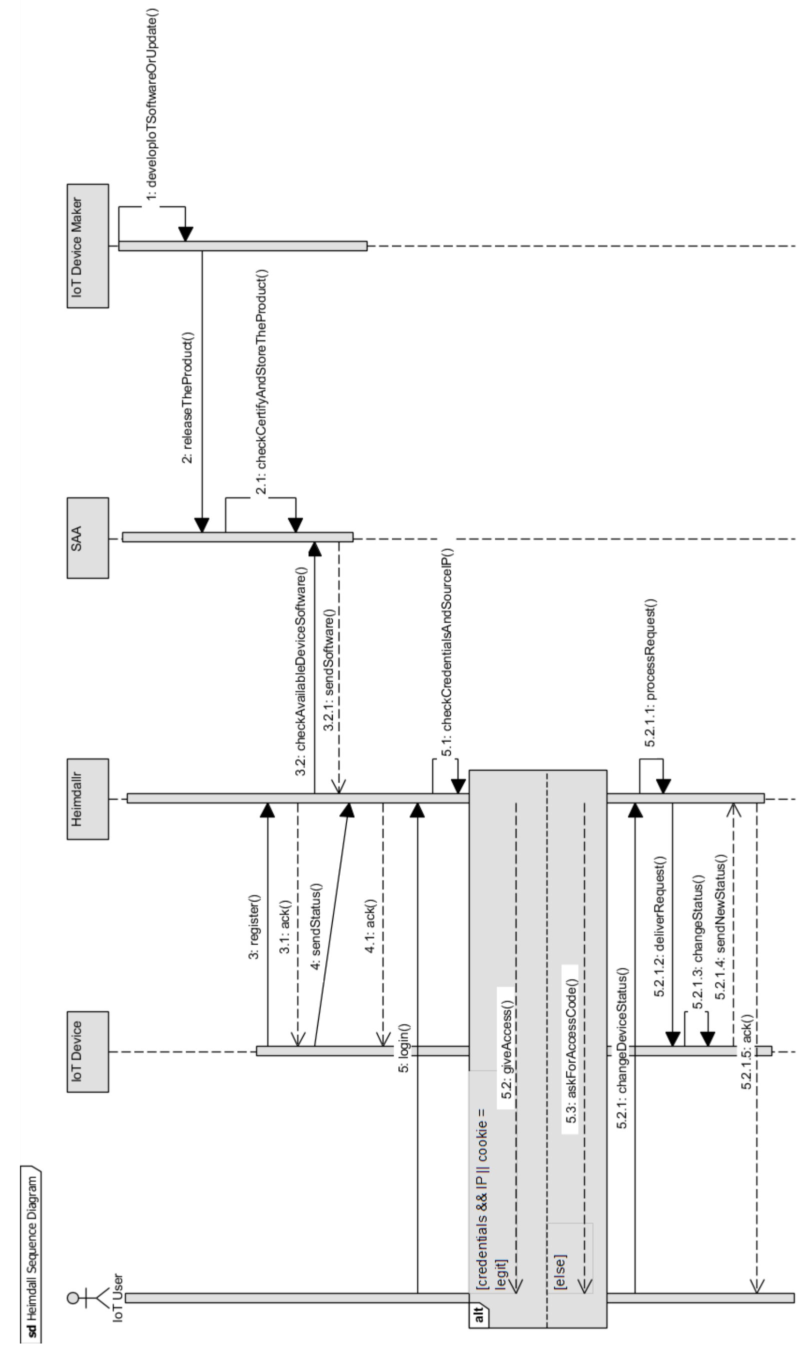
Figure 6: Activity Diagram of the Heimdallr System
This figure shows sequence of actions taken by the components of Heimdallr system in a particular scenario; basic user interaction with Heimdallr

In this figure, the scenario begins by *IoT User* taking a signal sending action called *Login.* Its corresponding received signal action is called *Check Credentials and Cookie.* Here *Heimdallr* checks if the entered credentials by the *IoT User* are correct or not. As it is explained in our use case diagram, there are three possible following actions. If every parameter received from the *IoT User* is valid and correct then access to *Heimdallr* is granted. Otherwise, either the user gets redirected to login page as a result of providing wrong username-password pair or *Heimdallr* detects a first time login (having no valid cookies) from a client-device and asks for an access code to proceed. In case of the latter one, *IoT User* takes the action called *Generate Code using Key Generator*. Meaning that, the user must use our key generator program installed on the client device and provide *Heimdallr* the generated code along with the correct username-password pair for a successful *Login*. Once the access is granted the user gets connected to *Heimdallr* and prompted the main page where all connected IoT devices are shown depending on the user's current role. Concurrently –in order to show user those IoT devices– *Heimdallr* takes an action called *Check Connected IoT Devices.* By doing so, it pings every connected device to trigger them to send their current status and version information. After receiving this data, aside from prompting user the main page where user sees his/her owned IoT devices, *Heimdallr* checks for any available update by querying *SAA*. There are again three possible outcomes of that action. First, everything can be up-to-date, and thus, we proceed without any action. Secondly, there can be a Heimdallr interface update for an IoT device or finally, there can be an update for an IoT device itself. In any case, those three possible actions merge by an or-join (a merge node) and wait to be joined by an and-join (a synchronization bar) as a result of the next *IoT User* action.

From the *See Connected IoT Devices* state *IoT User* can take three different actions. He/she can *Logout* and complete the scenario or *Change Settings* of *Heimdallr* such as the ones shown in Figure 5 or lastly, choose an IoT device to control. If he/she chooses the latter option, it gets joined with the awaiting action of *Heimdallr* –explained in the previous paragraph– by the and-join on the *IoT User* side and as a result the user can access the IoT device's page where the UI pertaining to that device is provided. This is the page where the user can create and send tasks to the IoT device through Heimdallr and see the changing status of it.

In the next subsection, the given sequence diagram will explain this exact same scenario in more detail by explaining some of the parallel actions taken by the system components but not shown in Figure 6.

4.1.4 Sequence Diagram



sd Heimdall Sequence Diagram
IoT Device Maker
SAA
Heimdallr
IoT Device
IoT User
1: developIoTSoftwareOrUpdate()
2: releaseTheProduct()
2.1: checkCertifyAndStoreTheProduct()
3.2: checkAvailableDeviceSoftware()
3.2.1: sendSoftware()
5.1: checkCredentialsAndSourceIP()
5.2.1.1: processRequest()
3: register()
3.1: ack()
4: sendStatus()
4.1: ack()
5: login()
alt
[credentials && IP || cookie = legit]
5.2: giveAccess()
[else]
5.3: askForAccessCode()
5.2.1: changeDeviceStatus()
5.2.1.2: deliverRequest()
5.2.1.3: changeStatus()
5.2.1.4: sendNewStatus()
5.2.1.5: ack()

Figure 7: Sequence Diagram of the Heimdallr System
This figure shows the interactions of the Heimdallr system components (elements) sequentially for a given time period

In the sequence diagram given in Figure 7 the interactions of the components (elements) of the system are shown in a timeline. From left to right the components are aligned and the period of time starts at the top and ends at the bottom. Thus, the very first action is taken by the *IoT Device Maker* by developing and releasing the product for an IoT device. That product can be any kind of software related to the IoT device produced by that maker (company). However, in our proof-of-concept demo it will either be a system update for an IoT device or an UI update of that device installed on Heimdallr. Note that, this is an asynchronous action that can be taken anytime by the *IoT Device Maker*, and this is why it was not shown in Figure 6; *Activity Diagram*.

Once the product is published, *SAA* receives and checks it and if it decides that the product has no security flaws threatening the data privacy, certifies and stores it to be sent when requested by *Heimdallr*. Meanwhile, an *IoT Device* requests registration to *Heimdallr* and *Heimdallr* checks if any service or update is available to download for or related to that device in *SAA* database. Right after the successful registration of the *IoT Device*, an *IoT User* requests login to *Heimdallr*. If user credentials are correct, the cookie sent in the HTTPS packet is valid and the IP address of the requester is not blacklisted *Heimdallr* grants the access. After that, user can take any action given in Figure 5. As it is mentioned previously, an action which changes the status of an IoT device is always delivered to that device as a task by Heimdallr. In Figure 7, we see that the *IoT User* takes an action called *changeDeviceStatus(),* this can be any such task explained in earlier diagrams. Note that, the previous action –called *register()*– taken by the *IoT Device* does not have to be followed by the *IoT User*'s login request, on the contrary, it is totally unrelated with user actions. What happens is that, once the user is logged in, only the currently registered devices will be shown on the main page which is nonetheless being continuously updated by Heimdallr.

The next subsections will present the lower-level system diagrams in an effort to provide a deeper understanding. We will first get an insight on the class diagrams of the system components which will be followed by the ER (entity-relationship) diagrams of the database entities used by *SAA* and *Heimdallr*.

4.1.5 Class Diagrams

In this subsection, class diagrams of all the classes implemented and used for our proof-of-concept system will be discussed in detail. Note that, in these diagrams some of the class variables, functions and constructors are not shown, as well as any graphical user interface (GUI) elements.

Except for the constructors, even though they have actually been implemented, unused functions such as some getter and setter functions, are also simply excluded from the diagrams for the sake of simplicity.

4.1.5.1 Heimdallr

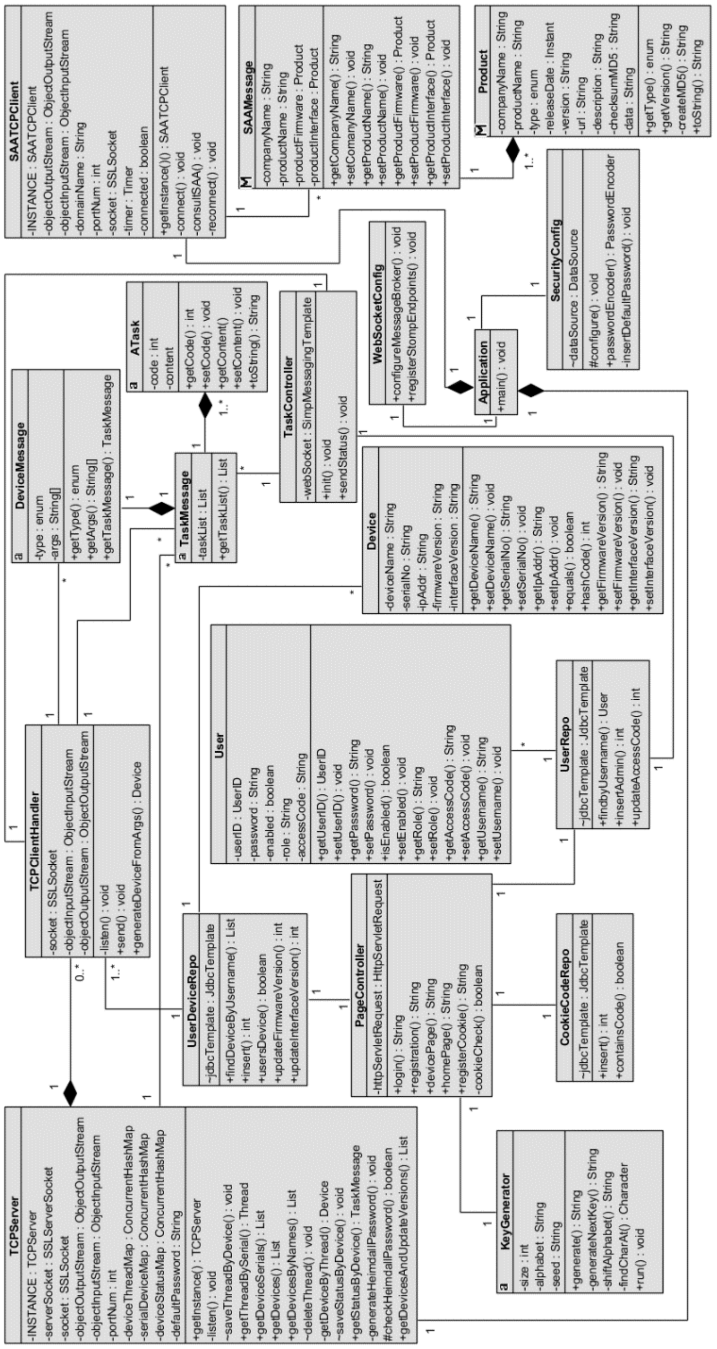


Figure 8: Heimdallr Class Diagram
This figure shows the relation between the classes of Heimdallr software; one of the main components of the Heimdallr system

In the class diagram for *Heimdallr* given in Figure 8, we clarify the different types of classes. Some of these classes go under the package called *models*, meaning that those classes are implemented to be used as entities of common objects among the components of the Heimdallr system. These are namely; *User, Device, ATask, Product, TaskMessage, DeviceMessage,* and *SAAMessage*.

User objects are used to hold data of *Heimdallr* users. Any user who is registered as either *ADMIN* or *USER* has a corresponding *User* object created and stored in *Heimdallr* database.

Device objects represent the actual physical IoT devices which have successfully established a connection to Heimdallr. Once an IoT device is connected, a corresponding *Device* object gets created with the device information received during the first stage of the connection; registration phase. It includes the device name, serial number, its assigned IP address, and installed firmware and interface versions. Note that for simplicity we used unique device name and serial number combinations as unique device IDs. Doing so we did not have to introduce MAC addresses when representing an IoT device. Moreover, usage of MAC addresses might also be problematic, more about that will be discussed in Chapter 5; *Simulation of Heimdallr and Experimentation*.

ATask object basically represents a task created and sent by Heimdallr to an IoT device. To generalize a task we introduced an integer code and a generic type content variable in *ATask* class. This code can represent anything and it is meant to be determined and configured by the *IoTDeviceMaker*, meanwhile, the content variable holds the data corresponding to that code. For instance, a task object with *code=1 : content=25* and *code=2 : content=true* could be sent to an IoT thermostat where code *1* means the temperature and code *2* means the power status of the device. Therefore, what the above task is trying to say is “turn on the thermostat and set the temperature to 25 degrees Celsius”. Since the content is of type generic, any IoT device can be controlled using the same class (*ATask*) having different interpretations upon receipt.

To send tasks from *Heimdallr* to *IoTDevice*s we introduced the class *TaskMessage*. A *TaskMessage* object consists of an array of *ATask* objects and is used to send tasks created by *Heimdallr* to *IoTDevice*s. Similarly, any message sent from an *IoTDevice* to *Heimdallr* is a *TaskMessage* object wrapped in a *DeviceMessage* object.

We introduced *DeviceMessage* class to be able to tell *Heimdallr* that the message it is receiving as a *TaskMessage* is either a *REGISTRATION* message or a *STATUS* message. Whenever an

IoTDevice tries to establish communication with *Heimdallr*, it first sends a *REGISTRATION* message along with *Heimdallr*'s password to be registered, then it starts sending its current status in time intervals determined by the *IoTDevice* itself.

Product and *SAAMessage* are similar classes used in the communication between *Heimdallr* and *SAA*. A *Product* object represents a product generated by the IoT device maker which will be explained later. It consists of the essential information required to describe it along with the binary data of the product itself. This information contains the company name which makes the product, product name, type, release date, version, URL which can be used to access the product's source, description of the product, binary data and the MD5 checksum of that data. For the purpose of our proof-of-concept version of the system, a product can be a *FIRMWARE_UPDATE* or an *INTERFACE UPDATE,* this is what we call the product type. Every time *Heimdallr* asks for an update, *SAA* responds back with an *SAAMessage*. An *SAAMessage* contains both the latest *FIRMWARE_UPDATE* and the latest *INTERFACE UPDATE.* Note that, in real applications sending the whole product every time it is queried is undoubtedly an inefficient practice. However, since we arbitrarily decided to have our products carry a mock-up data as small as 32 bits for test purposes, we didn't differentiate a respond with only the latest version information and a full packet respond carrying the complete product data.

Another package we called is *repository.* Classes called *CookieCodeRepo, UserDeviceRepo* and *UserRepo* fall under that package. Earlier we also called those classes DAO. DAO simply means Data Access Object. These classes are the links which connect entities mentioned above to their corresponding database objects. Therefore, in our case whenever a Java object has to be inserted into a database table, we call its corresponding DAO, give the object to it so that it can establish the connection to database and execute the SQL query to insert that given object. The same applies to every kind of interaction made with the database. Thus, *UserRepo* is used to store, fetch and check a *User* object to and from the user table in *Heimdallr*'s database. *CookieCodeRepo* is used to store and check if the cookie received from a client device is valid or not. Similarly, *UserDeviceRepo* is to execute any query related to users and the devices owned by them. One such query is carried out by *findDeviceByUsername()* function of that class. Given the name (unique) of the user, it returns a list of *Device* objects to be used by another *Heimdallr* task.

Now, we would like to separate the Heimdallr software into two ends and explain them independently. First one is the internal network. Here internal network means the communication between Heimdallr and connected IoT devices. And the other end we call external network meaning that the communication between Heimdallr and SAA, and any type of client device used to connect to Heimdallr.

For internal network, we used both *SSLSocket* and *SSLServerSocket* objects which come with *java.net* package to establish secure, full-duplex TCP connections between *Heimdallr* and *IoTDevice*s. *TCPServer* and *TCPClientHandler* are the responsible classes for that. This *TCPServer* singleton object always listens for *IoTDevice*s. Once a packet is received from an *IoTDevice*, it initiates a *TCPClientHandler* object specifically for that device and runs it on a new thread. From this point on, every interaction done with this *IoTDevice* is conducted and regulated by means of its corresponding *TCPClientHandler* object.

External network can also be divided into two parts. First part is the communication done with the SAA. This is similar to what we have done for the internal network communications. Instead of *TCPServer* class we have another singleton class called *SAATCPClient.* Again using *SSLSocket* it establishes a secure TCP connection to *SAA*. Note that unlike our internal network connections, this one is a half-duplex TCP connection, since *Heimdallr* is the only one which initializes queries and makes requests using *SAAMessage* objects. The second part however is different; communication between Heimdallr and client devices. *Heimdallr* provides a secure web service (using HTTPS protocol) that can be accessed from any internet connected device as it is described earlier. That is exactly why it needs to have a static IP address to function properly. Once a request made to Heimdallr's IP address it responds with its login page. After a successful login, clients are redirected to their homepages and so on (see previous diagrams). All those requests are handled by the *PageController* class. If a user clicks on an IoT device shown on the homepage, *PageController* redirects the user to the page (interface downloaded and installed from SAA) of that device. At this point the HTTPS communication between the client device and *Heimdallr* gets upgraded to WebSocket Secure (WSS) communication. After that every request from and to the client is handled by the *TaskController* class. To be more specific, any up-to-date status sent by an *IoTDevice* to *Heimdallr* is pushed to this device's page through WebSocket. Similarly any task created and sent by the client using that page is handled again by the *TaskController* and passed to the *IoTDevice* by means of *TCPClientHandler*.

In Figure 8, we can see a utility class called *KeyGenerator*. This class is responsible for generating new access codes asynchronously for users. These codes are then used to check the validity of the access code which will be entered by the user on the next attempt to login using a new client device. More about that will be talked in Subsubsection 4.1.5.5.

Finally, *Application* is the main class where the *TCPServer* and *SAATCPClient* are first initialized and all the configuration classes are bootstrapped, namely *WebSocketConfig* and *SecurityConfig*.

4.1.5.2 IoT Device

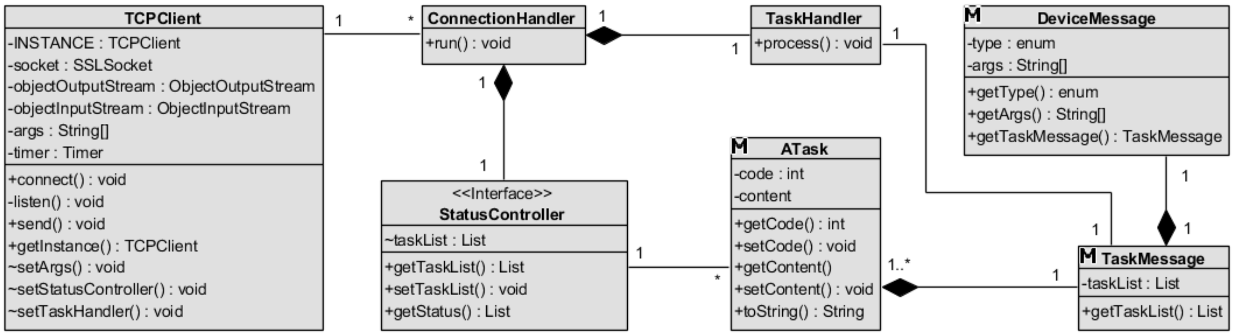


Figure 9: IoT Device Class Diagram
This figure shows the relation between the classes of an IoT device software; one of the main components of the Heimdallr system

The only other component involved in demonstrating the internal network working principle of the Heimdallr system is the *IoT Device*. In Figure 9, the class diagram of an IoT device is given. It is, however, essential to mention that this above program is meant to be used as the core library for creating variety of IoT devices such as an IoT thermostat or an IoT TV. To do so, one has to override the function called *process()* of *TaskHandler* class and implement the functions of *StatusController* interface. Hence, different IoT devices are given the same working principle which makes them compatible with Heimdallr while having unique and specialized ways of handling the tasks sent to them. *TCPClient* is the singleton class where the full-duplex secure socket communication over TCP is established with *Heimdallr*. *ConnectionHandler* is a helper class for the *TCPClient* and it is responsible for fixing connection issues and reestablishing it. For instance, if *Heimdallr* switches off *ConnectionHandler* notices that and starts pinging *Heimdallr* in preset time intervals. Once it receives a response from *Heimdallr* it establishes a new connection. Finally, *DeviceMessage, TaskMessage,* and *ATask* are exactly the same classes as the ones we explained in *Heimdallr* class diagram (Figure 8).

4.1.5.3 SAA

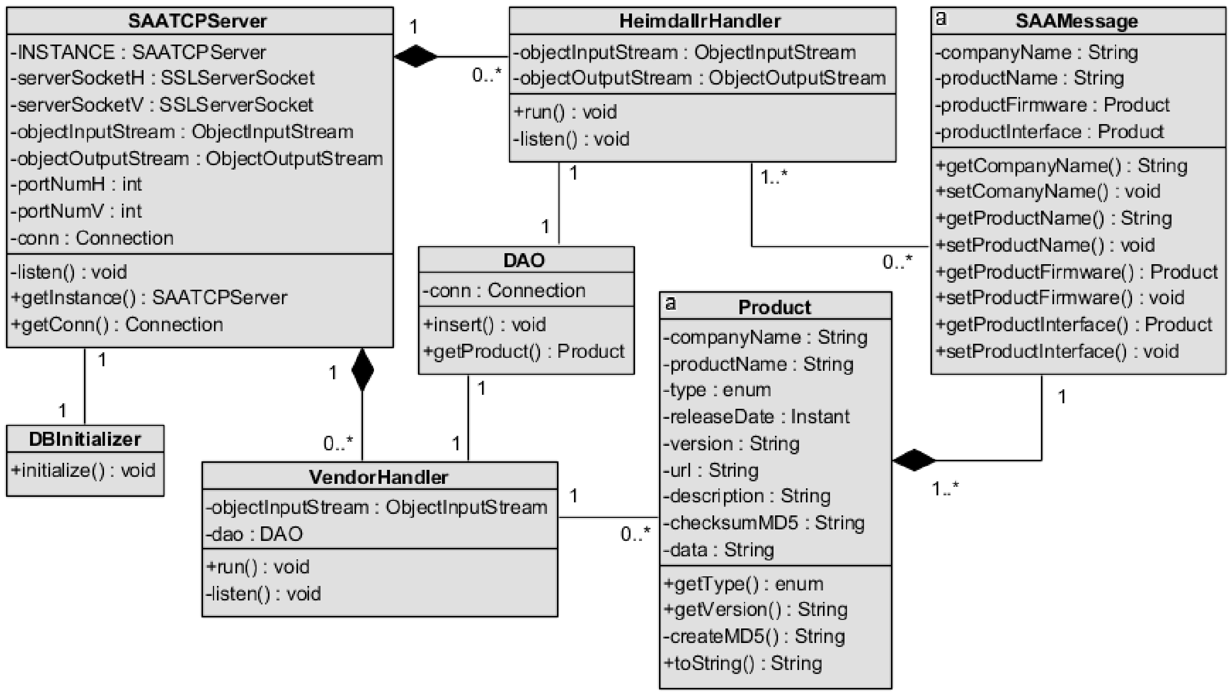


Figure 10: SAA Class Diagram
This figure shows the relation between the classes of SAA software; one of the main components of the Heimdallr system

On the external network side of the Heimdallr system, one of the three components is SAA. *SAA* represents the role of our not-for-profit organization SAA. It is composed of six classes. In the class diagram of *SAA* given in Figure 10, *Product* and *SAAMessage* are again exactly the same classes as we explained in *Heimdallr* class diagram (Figure 8). The main class of *SAA* is called *SAATCPServer*. It is responsible for establishing half-duplex secure TCP connections using *java.net SSLServerSocket*. It listens exactly two ports; one for *Heimdallr* connections and another one for *IoTDeviceMaker* connections. Any request received on port reserved for *Heimdallr* is transferred to and handled by the *HeimdallrHandler* class. Likewise, any request received on port reserved for *IoTDeviceMaker* is transferred and handled by the *VendorHandler* class. When a *Product* wrapped in an *SAAMessage* object is received from an *IoTDeviceMaker, VendorHandler* instantiates a *DAO* object and calls its *insert()* function to store the *Product* received into the appropriate table in the *SAA* database. Similarly, whenever *Heimdallr* makes a request asking the latest *FIRMWARE_UPDATE* and the *INTERFACE_UPDATE,* this time *HeimdallrHandler* instantiates a *DAO* object and calls its *getProduct()* function to fetch the latest updates from the database. Finally, although it is of no importance, *DBInitializer* is responsible for creating the necessary tables in the *SAA* database when it is first run.

4.1.5.4 IoT Device Maker

The second main component of the external network of the Heimdallr system is the IoT device maker. In our proof-of-concept application it is represented by the program called *IoTDeviceMaker.* Below its class diagram is given in Figure 11.

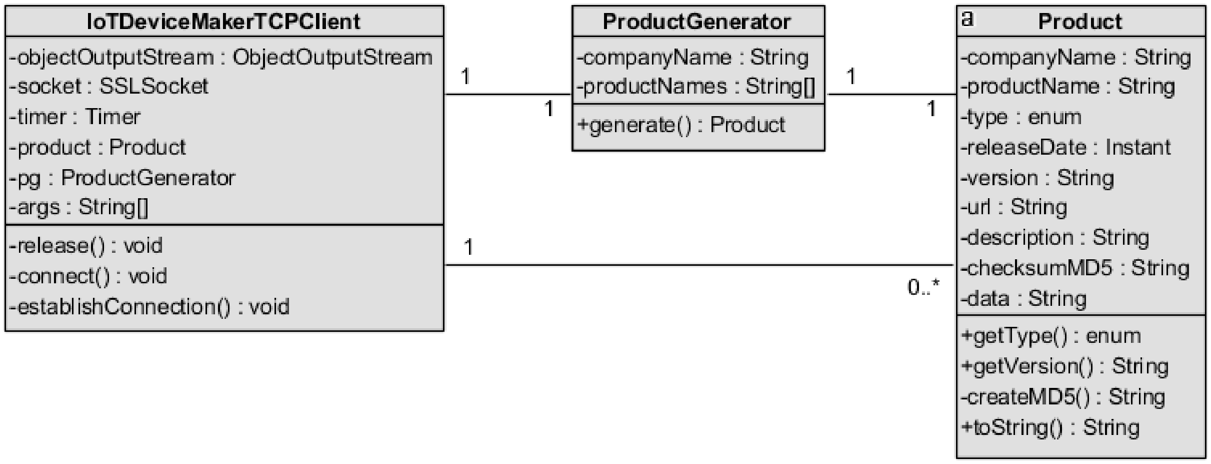


Figure 11: IoT Device Maker Class Diagram
This figure shows the relation between the classes of an IoT device maker software; one of the main components of the Heimdallr system

As always *Product* is the object representing either a *FIRMWARE_UPDATE* or an *INTERFACE_UPDATE* made by the *IoTDeviceMaker.* Here the role of establishing secure TCP connections is assigned to the class called *IoTDeviceMakerTCPClient.* This class is also responsible for the only task that an IoT device maker has in our proof-of-concept system; production and publication of an update. It instantiates an object of the *ProductGenerator* class using the arguments (*args[]*) given at run time. These arguments tell the *ProductGenertor* what kind of products it can produce under which company's name. Then it randomly generates every attribute ranging from *version* number to *data* required to create a *Product.* Once it is done *IoTDeviceMakerTCPClient* sends the product to *SAA* to be stored.

4.1.5.5 Client Device

The final component of the external network of the Heimdallr system is the client device. A client device consists of the key generator and a software which allows users to connect to the Internet e.g. a web browser. Since Heimdallr provides a web application service for its users, they will be interacting with Heimdallr by simply connecting to it over the Internet by means of no matter what. That is why in this subsubsection we will only be discussing the class diagram of the key generator (given in Figure 12) needed for generating access codes for the first time logins.

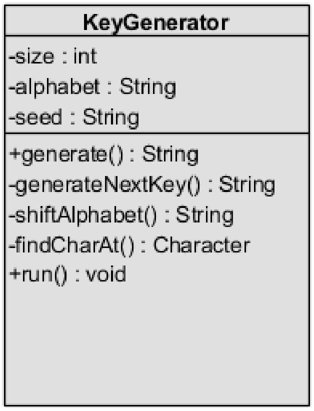


Figure 12: Key Generator Class Diagram
This figure shows the only class needed for generating access codes by the software installed on client devices. It is one of the main components of the Heimdallr system

The *KeyGenerator* is the class which asynchronously generates user specific, one-time-use access codes to be used when logging in for the first time to Heimdallr's web interface from a new client device. Having a *size, alphabet* and *seed,* it generates unique access codes every time the function *generate()* is called. It ensures the validity of these asynchronously generated access codes by working in tandem with the *Heimdallr*'s *KeyGenerator* mentioned in Subsubsection 4.1.5.1 and in deployment diagram Subsection 4.1.1. Both of these *KeyGenerator*s are exactly the same, and therefore, use exactly the same algorithm to produce and validate these codes. Let us explain how. On the client device the *KeyGenerator* gets initialized with a given *seed*. This is usually a predetermined password chosen by the user. When the public function *generate()* is called by the user, it calls the private function *generateNextKey()* and feeds it with the *seed.* This function first calculates the octal values of the characters composing the *seed.* It then gets the maximum of them and shifts the *alphabet* by this maximum value. Now that we have a new alphabet, it is time to generate a new access code! To do so, our algorithm iterates through the given *seed* one more time and replace each character with a new one from our new alphabet. Since both the server-side *KeyGenerator* and its counterpart client-side *KeyGenerator* have to generate exactly the same new code without communicating, there had to be a precise way of picking these new characters from our new alphabet rather than picking them randomly. Hence what our algorithm does is, it adds the index value of the current character of the *seed* to its octal value and uses the resulting integer value as an index to get a new character from our new alphabet. We then end up building a new String of the same length after iterating through the characters of the current *seed*. Finally we reverse this String and return the resulting value as the new access code and the next *seed* to generate the next code.

The counterpart of that algorithm in server-side does exactly the same thing once it receives and checks the validity of the current *seed* and updates the corresponding user entry in database by changing that user's access code with the newly generated one to be checked the next time. This is how we generate one time use access codes in an asynchronous and stateless manner.

4.1.6 ER Diagrams

In this final subsection of high-level system diagrams, we will be examining Heimdallr system's entity-relationship (ER) diagrams. This type of diagram shows the relationship between database entities and the way they are stored. Hence it helps us fully comprehend the system. In our proof-of-concept version of Heimdallr system there are only two components which use a database; *Heimdallr* and *SAA*. Recall from the class diagrams of *Heimdallr* and *SAA* that the DAOs or repositories were the classes that carry the data to and from their databases. Below in Figure 13 and Figure 14, ER diagrams of those databases in their third normal form (3NF) are given.

4.1.6.1 Heimdallr

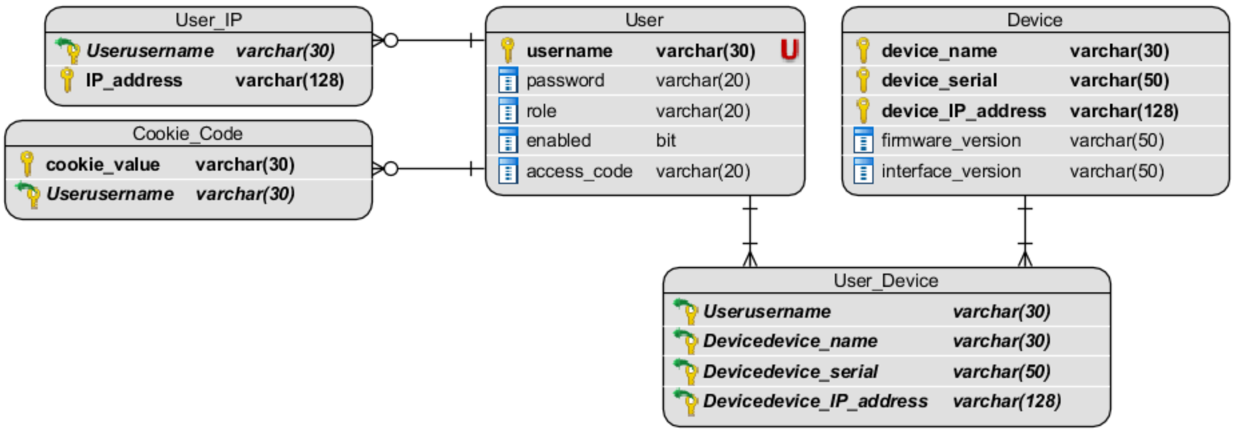


Figure 13: Heimdallr ER Diagram
This figure shows the relationships of entity sets stored in Heimdallr database

User objects are stored in *Heimdallr* database as *User* entities. A *User* entity must have a unique *username* attribute as its primary key. A *Device* entity, on the other hand, has a primary key which consists of three attributes; *device_name, device_serial,* and *device_IP_address.* These three attributes together are what make a device distinguishable from others. The relation called *User_Device* is a join table of *User* and *Device* entities. This allows us to easily match a device with its owner. Similarly, in order to comply with 3NF, *User_IP* and *Cookie_Code* tables are separated from the *User* table and put in a many-to-one relationship with it. They simply hold the data on IP addresses and cookies assigned to specific users.

4.1.6.2 SAA

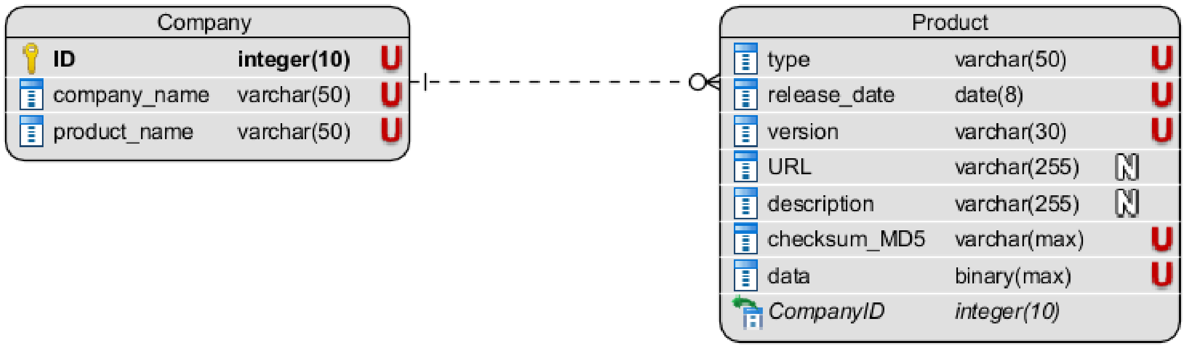


Figure 14: SAA ER Diagram
This figure shows the relationship of entity sets stored in SAA database

The other database in the Heimdallr system is used by *SAA*. A *Company* entity has a unique *company_name* and *product_name* combination represented by a uniquely assigned *ID.* This *ID* is then used to create a one-to-many relationship with the *Product* entity. A *Product* represents a product created by an IoT device maker and sent to SAA to be stored. When *Heimdallr* asks for the latest version of a specific *IoTDevice* by sending *SAA* the company name and the product name, *SAA* queries its database with these values to find any matching products. It first sorts the matching results in descending order by their *release_date* and group them by their *type.* Finally it returns two products, one as the latest *FIRMWARE_UPDATE,* and the other one as the latest *INTERFACE_UPDATE,* and sends them to *Heimdallr* wrapped in an *SAAMessage.*

4.2 Tools and Technologies Used

During the development of our proof-of-concept version of Heimdallr we used Java version 8 as the main programming language, and JavaFX to implement the GUI of the test components which will be presented in Chapter 5. Our first attempt was to demonstrate the system using Akka actors[10]. It was a single project composed of five actors; *HeimdallrActor*, *SAAActor, IoTDeviceActor, IoTDeviceMakerActor,* and *ClientActor*. However, despite that it was able to deliver the idea of how the system works, it was also way too simple and shallow to be able to show details on security issues or the interaction of a real user with the system. Hence, we gave up on actors and created four different *maven* projects, one for each component of the system. Implementation of *IoTDevice* and *IoTDeviceMaker* were straightforward. As we discussed in Subsection 4.1.5, they consist of a couple of simple classes which allow them to establish the necessary connection with other components of the system. Each of these connections between the system components –including the ones established by *Heimdallr* and *SAA*– are either half or full-duplex secure TCP connections achieved using *SSLSocket* and *SSLServerSocket* both are found under *java.net* package. Since these classes provide secure connection via sockets and server sockets using protocol called Transport Layer Protocol (TLS) or formerly called Secure Sockets Layer (SSL), we needed our own certificate to be shared during the TLS handshake. Therefore, we created two self-signed certificates one for *SAA* and another one for *Heimdallr* using Java Keytool (see Appendix A).

[10] A free and open-source toolkit which provides programming using the actor model. The actor model provides an abstraction that allows the programmer to model the classes as stateful entities communicating with each other asynchronously by explicit message passing [88].

Although the type of those certificates is of no importance in our proof-of-concept version, we used *SHA256withRSA* signature algorithm to sign them. Additionally, we only used TLS version 1.2, since version 1.3 seemed to have some bugs which would unnecessarily obstruct the goal of the project. And to be compatible with our certificates we only used cipher suit "*TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256*".

The biggest and the most complex component of the entire system is the *Heimdallr* itself. Since it was such a complex program to implement, considering our limited time we decided to use Spring Framework and let it eliminate the boilerplate configurations needed to startup our web application. As we mentioned earlier in this paper that there are two parts of Heimdallr software that we can name; internal and external network. We know that the internal network communication is achieved by secure Java sockets. External network however, is where the Spring Framework comes into play. We used Spring MVC to build *Heimdallr*'s web application which allows users to access their Heimdallr, and therefore, their IoT devices. MVC architecture composes of three main components Model, View, and Controller. As we have discussed in class diagram of *Heimdallr,* its models are simple POJOs (Plain Old Java Object); *User, Device, ATask, Product, TaskMessage, DeviceMessage,* and *SAAMessage*. Its controller that handle user requests is the *PageController*. And finally its views (pages) are JavaServer Pages (JSP) which are dynamically generated web pages on the server-side.

We mentioned that the communication between the client device and the IoT device's UI provided by *Heimdallr* is achieved using WebSockets (WS). To do that, we benefited from Spring Framework's STOMP support. Simple Text Oriented Messaging Protocol (STOMP) is a simple HTTP-like messaging protocol which is widely used in WebSocket applications. In our case, when a client device accesses an IoT device's interface (page), it immediately tries to subscribe itself to the *Heimdallr*'s WebSocket channel using SockJS which is essentially a JavaScript library that allows us to create a communication channel by providing a WebSocket-like object. Meanwhile, our *WebSocketConfig* class shown in Figure 8 acts as the STOMP broker to the client. Then, any message (task) sent to that channel is routed to *Heimdallr*'s *TaskController* to be handled accordingly. Hence, since it is a full-duplex bi-directional channel, client device does not require a refresh to see changes on the device page. Moreover, in order to make *Heimdallr*-client device communication absolutely secure we forced client devices to use only HTTPS protocol instead of HTTP and WSS instead of WS by one more time utilizing *Heimdallr*'s self-signed certificate.

The other two essential Spring Framework extensions that we have used were Spring Security and Spring Boot. We have used Spring Security to configure *Heimdallr*'s login page and login functionality, and to give or deny access to some URIs (Uniform Resource Identifier) depending on the roles and credentials of the users. And Spring Boot allowed us to run *Heimdallr*'s web application on an embedded Tomcat server that comes with it.

Finally, before we end this section, we would like to mention our database choices. Although any other type of database can be linked to our DAO classes at any time, now, for the sake of simplicity and to make our demo lightweight we used H2 –an in-memory database– for both *Heimdallr* and *SAA*. The only difference though, *SAA*'s DAO uses *PreparedStatement* class found under the *java.sql* package to query its database, while *Heimdallr*'s DAOs use *JdbcTemplate* class found under the *org.springframework.jdbc* package. However, during our tests we have realized that under heavy load *PreparedStatement* has a faster response time than *JdbcTemplate*. Hence, this is something that needs to be thoroughly examined and tested during the development of the real product.

4.3 Conclusion

Implementation of the Heimdallr system, while intuitively simple, requires complex acceptance issues which are beyond the realm of this thesis which introduces a concept with its proof-of-concept simulation. Aside from its exigent requirements for this idea to work, the intended features for Heimdallr needs a team of dedicated developers, fund, and most importantly time. However, to validate our concept, we have implemented a dry-run-test version of the proposed system to be tested and evaluated to see its competency.

To sum this chapter up, we would like to underline that the tools we have used and the design decisions we have made are as we have discussed in this section, strictly given that we are creating a proof-of-concept simulation of the Heimdallr system. There is no doubt that some of those design decisions such as usage of in-memory databases need to be changed along with possibly some technologies used for this project. In the next chapter we will be examining the simulation we have created by providing an intuitive flow diagram which will help visualize the overall Heimdallr system and we will discuss the experiments we have made on it.

5 Simulation of Heimdallr and Experimentation

In the following intuitive flow diagram (Figure 15) we provided the total of the system components and their relationships. In the diagram, straight lines indicates that the two components linked by that line, are somehow connected either by having a TCP connection, by being the same process, or by running in the same machine. Meanwhile, dotted lines indicate the flow of process by GUI elements. For bigger version of the parts of Figure 15 please refer to Appendix B. Note that, in these figures “S” and “L” are the arbitrary IoT device maker company names.

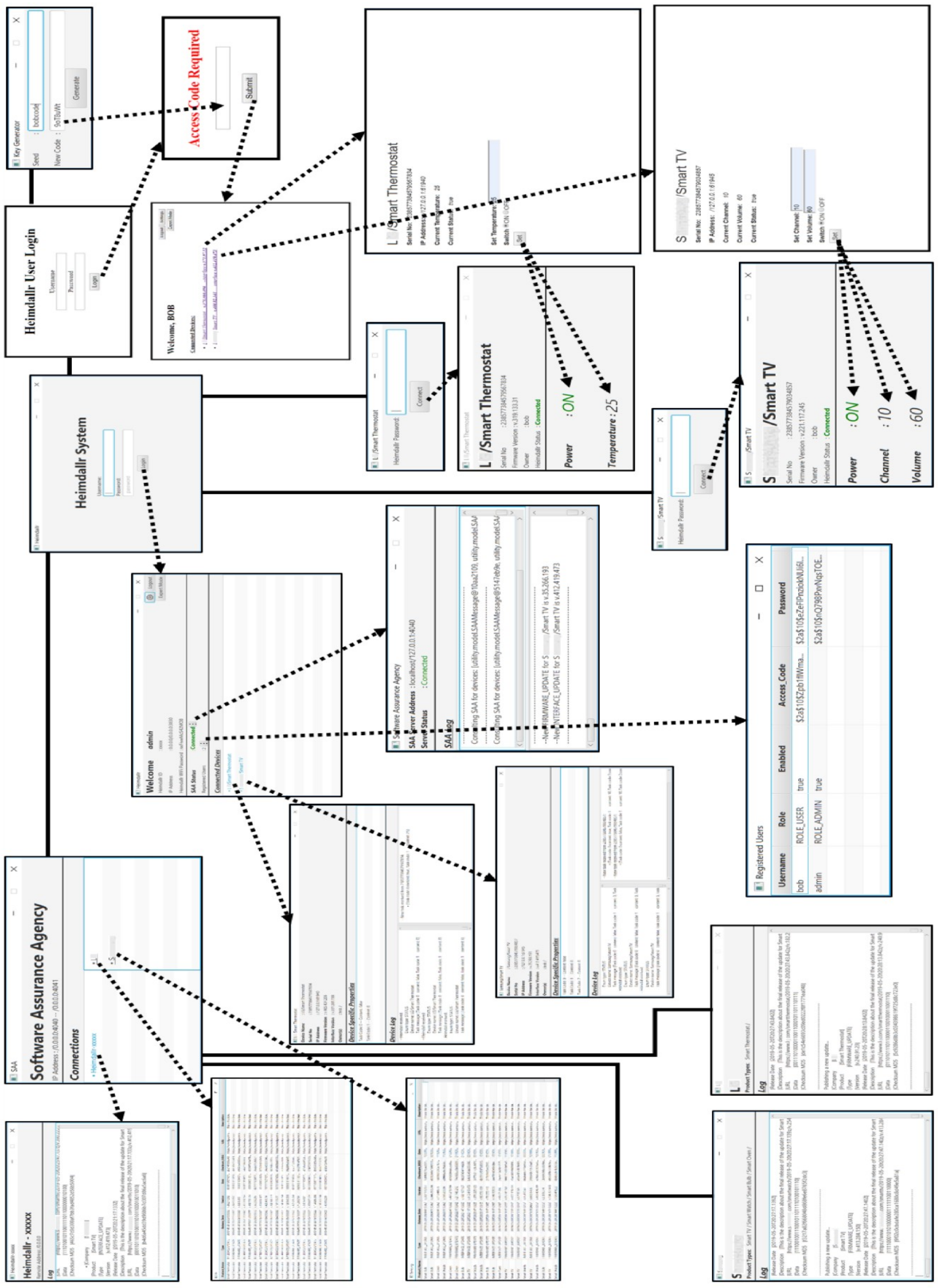


Figure 15: Intuitive Flow Diagram of the Heimdallr System
This figure provides an overall view for the Heimdallr system simulation

As we can see in Figure 15, we have a total of 8 major system components for the simulation of the Heimdallr system linked to each other with bold straight lines. These are, the Heimdallr software, the SAA software, a web browser, a Key Generator, a number of IoT device producers, and their IoT devices (here we have limited to two producers with one IoT device from each), all running on a single system. We set one of the IoT device makers to simulate the company "S" and the other one to simulate the company "L". Likewise, one of the IoT devices is set to be a S's Smart TV, while the other one is set to be an L's Smart Thermostat.

Both S and L companies are connected to SAA and any update produced by them is sent to SAA which can be seen in their *Log* section. We can also see what kind of products they make underneath their title. Similarly, from SAA window we can access all of the connected IoT device makers' products sent so far and stored in SAA database. We can also access any Heimdallr connected to this SAA and their log under the *Connections* section. Note that, in our simulation we only have one Heimdallr active at a time with an ID; *xxxxx*.

In the diagram, we see that Heimdallr has an admin interface which leads the admin to a home page after a successful login. There, the admin can see the *Connected Devices, SAA Status,* and *Registered Users.* A connected IoT device's page gives details about the device such as the device name, serial number, currently assigned IP address, current firmware and interface version, and the owners' username. In the middle section of its page, *Device Specific Properties* are given. These are basically the device specific task codes designated by the device maker and their current content values. Finally, a device page provides *Device Log*s at the bottom section. There, the log of any task sent and any *DeviceMessage* received by the Heimdallr is shown.

The connected SAA's page can be accessed via the button provided next to *SAA Status* on the main page of Heimdallr admin interface. This page shows two log sections. The above one gives any message sent to the SAA and the bottom one gives any response received in return. And finally, an admin can see the table of currently registered users of this Heimdallr along with their information.

As we said, in this simulation we have two connected IoT devices. One is L's Smart Thermostat and the other one is S's Smart TV. In order to connect these devices, one needs the Heimdallr password created for device registration. Once the correct password is entered, these devices get connected to our Heimdallr as it is shown in Figure 15. We also created an interface for the application simulating an IoT device. This interface pops up once the device is connected to Heimdallr. There we can see the device and its maker's name, serial number, current firmware version, owner's username, Heimdallr connection status, and finally on the bottom section their device specific properties which is exactly what we see in *Device Specific Properties* section in Heimdallr interface's device page.

Last component of our simulation is the web interface of Heimdallr which is used by the users to interact with their IoT devices connected to it. In Figure 15, we see the login page of this interface with a title of "Heimdallr User Login". After entering the right credentials, if there is no stored valid cookies in this client device, the user can be prompted to a second page where an access code is requested. Using the Key Generator, the user can generate a new code and proceed to the user's home page. Here, the user can see his/her owned and currently connected IoT devices and access their interfaces to interact with them. Every, device interface is specifically created by the IoT device maker and certified by the SAA. The latest certified version is regularly checked by Heimdallr and installed when a new one is available. We see that any task sent by the user through the web interface of Heimdallr is delivered to the actual IoT device.

In the rest of this chapter we will be explaining what kind of experiments we have made using this Heimdallr simulation and what is our overall assessment of the system.

5.1 Experimentation

Prior to begin developing the Heimdallr system, in order to make ourselves believe in our cause before anyone else, we wanted to make sure that the security issues of IoT devices are indeed as serious and ubiquitous as we have read from countless articles. Hence, we have discovered a website called Shodan [81] where IP addresses of all of the internet connected IoT devices are listed. Here we browsed for IP cameras and picked the URLs of five of them. We also searched for default username-password pairs for those cameras by their manufacturer and the product names. Then, we started trying those pairs out which we have found on the Internet for free and publicly! The results were actually shocking, we managed to access interfaces of three IP cameras out of five with ease. The worst part is, we could even change the angle of the cameras, get live stream and even change settings and none of it was even illegal. Now that we had incited, we started the development process right away.

With a working proof-of-concept simulator of Heimdallr system, we first checked the integrity of the system to make sure that there were no flaws within the basic functionality including its database related actions. Every use case is tested with a single user and single IoT device, single user and multiple IoT devices, multiple users and single IoT device, and multiple users and multiple IoT devices using different client devices with different cookie and IP addresses. We also tested *SAA* with single and multiple *IoTDeviceMaker*s with single and multiple types of products.

Since one part of the *Heimdallr* is a web application, the system must be robust enough to be able to negate every kind of known malicious web attack for absolute security. However, since penetration testing is completely another branch of expertise, due to our time limitations, at this point we have only focused on and avoided some of the most common attacks on web services, namely Cross-site Request Forgery (CSRF), SQL Injection, and Cross-site Scripting (XSS). Although CSRF protection is already provided by the *CsrfConfigurer* function of the Spring Security by default, we had to test XSS and SQL Injection attacks on our web application to make sure that it has no security holes on such a big scale.

After a thorough examination of the Heimdallr source code, we could only find one function where an SQL Injection might actually be harmful; Heimdallr admin interface authentication. In this function we feed the injected data source –which is the Heimdallr database– to Spring configured *AuthenticationManagerBuilder* (*auth*) along with the username and password pair entered on the login page which then executes the following code:

```
auth.jdbcAuthentication().dataSource(dataSource)
        .usersByUsernameQuery("SELECT USERNAME, PASSWORD, ENABLED FROM
        USER WHERE USERNAME=?")
        .authoritiesByUsernameQuery("SELECT USERNAME, ROLE FROM USER
        WHERE USERNAME=?");
```

Since, this authentication process handled by the Spring Security we were not able to change the source code. Therefore, in order to make sure that it is resistant to SQL Injection, we ran the attack using our own malicious SQL queries as username and password inputs. Some of them are as follows:

- *;DROP TABLE USER;*
- *" OR ""="*

- *admin OR 1=1*

And similar variations and combinations of these, which eventually return true statements when executed. Fortunately, we were not able to obtain any malicious results from our attacks. Thus, this experiment has proved that the Heimdallr software is resistant to SQL Injection attacks, and left us with the third attack which we need to avoid; XSS.

While we were examining the source code of Heimdallr to find any vulnerability to SQL Injection attacks, we also kept an eye for the flaws which may be vulnerable to XSS attacks. Eventually, we have noticed that the task messages (*TaskMessage*) created by the user on the client device and sent to the corresponding IoT devices via Heimdallr, are never checked. Therefore, any malicious script code can be stored as the content of a task (*ATask*) and sent to the IoT device as a task message. We also realized that if the IoT device accepts that task and stores it as its status, it eventually sends it back to Heimdallr to be displayed as the current status of this device on the web interface. This was the biggest vulnerability of our system, since, to some extent it relies on cookies when it comes to web application security, and stealing cookies is one of the most common use cases of XSS attacks. If a perpetrator hijacks a Heimdallr user's session by stealing his/her cookie through an injected script, any IoT device owned by this user becomes exposed to further malevolent actions. To patch that security hole, we added a small code which utilizes a library called Jsoup and strips any potentially harmful character or HTML tag from the task message created by the user before sending it to the corresponding IoT device:

```
task.getTaskList().replaceAll(aTask -> {
   if(aTask.getContent() instanceof String)
      aTask.setContent(Jsoup.clean((String)aTask.getContent(), Whitelist.none()));
   return aTask;
});
```

By doing so we assured that any input received from the user who uses Heimdallr web interface, is cleansed from any potentially harmful content, and therefore, safe to be sent and stored. In the next part of the development work, our goal is to test all other known attacks and address any vulnerabilities found in Heimdallr software.

Aside of these security patches, we also made sure that Hypertext Transfer Protocol Secure (HTTPS) is the only protocol used by the client devices when communicating with Heimdallr to increase security. However, during our tests we realized that the web antivirus programs and the browsers themselves are blocking the access to Heimdallr because of the fact that its TLS certificate is self-signed. Thus, for the real product we have to get a trusted CA (Certificate Authority) approved one. Furthermore, it is important to realize that since quantum computers started to become an actual tool today, as most of the DLT algorithms does, Heimdallr should also adopt quantum immune encryption algorithms rather than just relying on algorithms like RSA (Rivest–Shamir–Adleman).

We also tested the consistency of our asynchronous *Key Generator* algorithm to make sure that the two-factor authentication works flawlessly when a request is made from a new client device that doesn't carry a valid cookie. Although, the current algorithm works as intended, we noticed a small bug that can lower the security level that we are trying to achieve and maintain. The alphabet that we are using to create the next access code for a user is always the same every time. Moreover, since every user uses the same key generator algorithm, everyone's generator uses exactly the same alphabet to generate their codes. We know that if the first code assigned by a user is known, the next one for this user can be and is generated by feeding it to the key generator. Which means that, any time in the future if a perpetrator reveals the last used access code of a user, this person can effectively generate the next code using his/her own key generator without the actual owner's permission and knowledge. In order to prevent that, one of our primary goals is to introduce one more integer seed picked by the user and save it along with the user's first code in both Heimdallr and user's own key generator. This seed will then be used to scramble the alphabet every time a new access code needs to be generated. Furthermore, this scrambled alphabet will also be stored behalf of that user to be re-scrambled during the next key generation process. Therefore, even though a perpetrator was able to obtain the last code used by the user, in order to generate the next correct key, he/she has to use the exact same key generator application previously used and owned by the user.

Furthermore, during our tests, we also noticed some other misthought features that we have implemented were either broken or posing a security risk. For instance, at first we were using MAC addresses when registering an IoT device to Heimdallr. However, since MAC addresses are never sent encrypted, an attacker could eavesdrop on the packets that is sent from a wirelessly connected IoT device and through MAC spoofing, an unknown IoT device can impersonate that registered device and gain access to Heimdallr. Since, one has to physically access a device to get its serial number, we decided to use it along with the device’s name and Heimdallr password when registering to prevent MAC spoofing and related attacks. Another example, was the feature that allows users to blacklist or whitelist IP addresses to indicate whether to allow or deny requests originating from them. However, we obviously missed the fact that only Heimdallr’s IP address is static and almost all of the client devices have dynamic IP addresses. Thus, blacklisting a client device’s IP could only help users to reject any request from that device until its IP address is reassigned. Therefore, we removed that feature and then introduced cookies. Though, it is still possible to blacklist IP addresses to restrict Heimdallr to not receive but send data to those destinations. Doing so it could actually help preventing unwanted data exfiltration or DDoS attacks launched with the Heimdallr to some extent.

One more important observation to mention is the somewhat high traffic on *SAA*. When we were analyzing the network traffic of *Heimdallr* we have noticed that the *Heimdallr* is actually exhausting *SAA* by continuously requesting for new update information in predetermined time intervals. When a real life scenario is considered where thousands of Heimdallrs requesting their regional SAA servers, it does not sound good. Hence, our proposition to that issue is to assign unique IDs to each and every Heimdallr and let them register themselves to their regional (the closest) SAA server. So, instead of letting Heimdallrs to do the requests, we can have SAA servers to notify their registered Heimdallrs when there is actually a new update to be installed. This would definitely lighten the traffic on SAA servers, moreover, these unique Heimdallr IDs can be used to implement even greater features in the future.

Overall, aside of these small tweaks, the current version of the Heimdallr system works as intended with all the basic functionality after all of the tests on which we ran on local network. However, even though the idea of this system showed its competency to deal with the current IoT related issues in our tests, we concluded that for more accurate results, an alpha prototype should be implemented and tested in a typical real usage replication with a mix of simulated smart IoT devices to actually see how it performs in practice.

5.2 Analysis and Assessment

In this final section of this chapter, we would like to discuss how hard or easy is to realize the Heimdallr system and what the technical requirements are to put it into practice. In terms of IoT devices, IoT device makers and client devices nothing is different. Only exception is that IoT device makers will have to follow a standard when developing software for their IoT devices. This standard as we discussed earlier is what allows Heimdallr to locally run both the server-side services and the client-side applications of IoT devices which provide its users a UI to interact with. We also said that these applications or UIs are currently composed of a couple of JavaScript and JSP files per IoT device in our proof-of-concept version. However, this can surely change to a point where Heimdallr has its own format and Application Programming Interface (API) for IoT device makers to use when developing Heimdallr compatible software.

Another must have is the usage of IPv6 protocol. In Chapter 3, Section 3.2, we have discussed that the Heimdallr needs a static IP to be able to inherit a client-server model. When the inevitable depletion of IPv4 address pool is considered, starting up a new, IP address hungry system like Heimdallr hinging upon it is just not wise. Therefore, an utter solution would be to assign IPv6 addresses to Heimdallrs which is fortunately possible.

When it comes to Heimdallr itself there is also nothing discouraging. We tested our system in a machine with the following specifications:

• Processor	: *Intel® Core™ i7-4700HQ CPU @ 2.40GHz*
• Installed Memory (RAM)	: *16,0 GB*
• System Type	: *64-bit Operating System, x64-based processor*
• OS	: *Windows 10 Home*

During our tests (see Appendix C) the highest CPU usage was 20% which occurred when *Heimdallr* first initializing the embedded Tomcat server. On the other hand, average CPU usage was around 0.2% with seldom spikes of 10%. When it comes to memory usage, the allocated JVM heap size was around 0.50 GB out of 4.26 GB and the heap used was always less than 0.25 GB throughout an 18 minutes test run. This 0.25 GB (250 MB) includes our H2 in-memory database (~2 MB), embedded Tomcat server (~15 MB), Heimdallr source code (~40MB), and a couple of JSP pages which have a size of less than 10 KB in total. Here a couple of things can change. For instance we may want to use a heavier web server like WebLogic which currently has a size of 800 MB or we may want to switch to a persistent database like MySQL which is around 300 MB right now. Therefore without any user or IoT device data, Heimdallr software can be estimated to have a size of around 300 MB minimum and 1.35 GB maximum. We also estimated how much space the database could occupy in practice. If we consider a home network with 10 users each having distinct 20 IoT device and 10 client devices with valid cookies and IP addresses, it makes a total of 10 tuple of user data, plus 200 tuple of device data, plus 100 tuple of user-cookie and another 100 tuple of user-IP data in our database. We generously calculated every tuple by ignoring the size of binary, int and enum type attributes and assuming every attribute as *VARCHAR(255)* (see Figure 13 for actual value and types). If we use UTF-8 encoding, every character can have a size of up to 8 bytes each which makes 765 bytes per attribute. At this point we can even be more generous and consider every attribute 1 KB. Therefore, for IP and cookie tables we have 2 attributes each, and for user and device tables we have 5 attributes each which makes 1450 KB in total in our case. Moreover, if we have 200 distinct IoT devices that means we have 200 distinct JSP pages with their corresponding JS files. We already know that the combination of these files is about 10 KB in our proof-of-concept scenario which makes an extra 2000 KB of data to be stored in our database. Hence, we can confidently say that under the given circumstances the estimated maximum size needed for the database would be 3450 KB or about 3.5 MB in total. When we consider the storage needed for Heimdallr software itself, 3.5 MB of database entry becomes insignificant for the result.

The final requirement needed to be considered is the bandwidth. Since one part of Heimdallr is a web application, it needs to have enough bandwidth to be able to maintain a fast and functional communication with its users. Since we analyzed the network traffic of our proof-of-concept version –running on localhost– using a loopback address, our approximation was highly rough and hypothetical, albeit, it should give enough information to move on. If we are to continue with our previous example where we had a home network of 10 users, the highest traffic can be expected when or if all of these 10 users request to access their IoT devices' UIs at the exact same time. Then Heimdallr has to send all those JSP and JS pairs to the requesters which will cause a spike in traffic. We said that a pair can have a size of 10 KB. Moreover, our analysis showed that the highest value of cumulative bytes recorded for a user was indeed around 9 KBps. However, let us push it to the limits and assume that our pair is 64 KB which is the maximum size a JSP file can be. Thus, if we assume that the Heimdallr in this house has an upstream bandwidth of 80 Mbps, it means that it can send 80000000 bits per second. Since 64 KB is equal to 512000 bits, this Heimdallr can send 156 UIs per second or in other words it can respond to 156 users at the same time under its heaviest traffic which is more than enough when considered that it has only 10 users. The same applies for the downstream bandwidth needed to communicate and fetch software updates from SAA. Hence, for a regular home network a network bandwidth of 80 Mbps seems to be more than enough for Heimdallr to function properly.

Considering all of the above approximations we envisage that an optimized Heimdallr software can easily run on a small single-board computer like Raspberry Pi 3 Model B+ [82] with the following specifications:

- *Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz*
- *1GB LPDDR2 SDRAM*
- *2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE*
- *Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)*
- *Extended 40-pin GPIO header*
- *4 USB 2.0 ports*
- *5V/2.5A DC power input*
- *Power-over-Ethernet (PoE) support (requires separate PoE HAT)*

It is however important to note that these approximations were only calculated for our proof-of-concept version of Heimdallr software. Hence, there is no doubt that the requirements of the real product will vary and most probably will be higher. Nonetheless, we do not foresee any unsolvable drawbacks or obstacles which can come along with the development of the real product.

6 Conclusion and Future Work

IoT devices are proliferating and becoming every day tools for the ordinary person. The ubiquity of these devices allows us to see them even in the private moments of our lives. The access of the data that we provide to IoT device makers –by using these devices– exposes our habits, routines and schedule in addition to the base data given up in good faith. Thus, all the security issues related to data exposed to the IoT device suppliers, the third parties they use, and the vulnerabilities hidden in the IoT devices' hardware and software is becoming a more serious threat to privacy. As a result of our study of those issues we realized that as long as an IoT device is able to access the Internet freely, privacy of the user data is compromised. For this reason we created the Heimdallr system. Heimdallr can be described as a guardian that enhances the function of the users' home routers and builds a fortress on private networks. It monitors and controls the incoming and outgoing data traffic to and from the IoT devices connected to it and only allows requests from trusted sources with authorized users having secure credentials. By doing so, regardless of whether there is a hardware or software flaw or an issue with a service used on server-side, it guarantees that the user data sent to and from the IoT devices never exfiltrate to any unwanted places. Additionally, it is able to connect to SAA for access to all certified software and its updates for an IoT device. The system thus guarantees that the users' sensitive and private data flows only between the IoT device and the user itself.

Our current goal is to implement an actual alpha version of the Heimdallr system and test it in a real life scenario. This can be done by modifying an already existing open-source router firmware such as DD-WRT which is a Linux-based open-source firmware developed as a replacement for the inconsistent stock firmware of some specific router models [83]. Moreover, as we discussed in this paper that the Heimdallr is a large scoped solution sitting in the hearth of the data traffic. Using this as an advantage, the system can be further improved by adopting an open-source IDS (Intrusion Detection System) which enables it to recognize patterns of more complex malicious attacks and blocks them. A candidate for such a task can be the open-source, network-based IDS called Snort [84].

Similarly, since all of the user data passes through Heimdallr, it can collect and store all that data for a later reference while having all of it under the control and possession of the owner of the system [44]. An example for such a reference could be data laboring as we discussed in *Proposed Solutions* section of Chapter 2. If and when this kind of an idea is put into practice, Heimdallr users will be able to consider selling their stored personal information to whom paying for it. There are also some other ideas and small tweaks waiting to be realized to make the system more reliable and secure. An example can be the development and usage of a stand-alone application running on client devices and linked to each user's own Heimdallr with a one-time password and/or access code. That way no other connection can even be set to anyone else's Heimdallr. Additionally, the IP address of the requester can automatically be blacklisted by the system after a couple of wrong username-password tries. Even though, these are just small tweaks on the system, they can make a big difference against some specific type of malicious attacks. A major addition to our proposal, however, can be the provision of public not-for-profit services for IoT devices to use. If we want user data to be utterly safe and kept private without giving up on some essential third-party services which are required for some IoT devices but cannot be provided locally, similar services must be established by the trusted authorities. An example of such a service could be weather forecast. If an IoT device requires this kind of a service to be fully functional, it has to be made available but not by a third-party.

It is also important to realize that a system like Heimdallr may be very difficult with the current technology to put into practice because of its need of static IP, high performance hardware, and the system's must have; SAA. For this reason, we suggest that it can first be implemented in small-scale, for instance, for VIPs (Very Important Person) or military personnel and then made public for regular users over time.

As our final thought, a system like Heimdallr can be the foundation of greater developments and breakthroughs in distributed technologies while assuring the data privacy. An example idea can be a Facebook like system created by mesh of Heimdallrs exchanging some personal information under its owner's consent. This is just a simple example of what can be achieved in the future with the help of such a system.

We see our system as the start of the swing of the pendulum to the era where the data subject did not need outside for-profit data storage and service (cloud) and would lead to other development to re-create privacy preserving version of what is being offered by today's big tech companies.

With judicial user interface, operation of Heimdallr would be made as easy and convenient as some of the better mobile phones; suitable for the tech non-savvy!

7 References

[1] C. Group, CSA Group, [Online]. Available: https://www.csagroup.org/. [Accessed 6 March 2019].

[2] U. LLC, "About UL," UL LLC, [Online]. Available: https://www.ul.com/aboutul/. [Accessed 6 March 2019].

[3] K. Ashton, "That 'Internet of Things' Thing," RFID Journal, 22 June 2009. [Online]. Available: https://www.rfidjournal.com/articles/view?4986. [Accessed 18 March 2019].

[4] J. Clark, "What is the Internet of Things?," IBM, 17 November 2016. [Online]. Available: https://www.ibm.com/blogs/internet-of-things/what-is-the-iot/. [Accessed 18 March 2019].

[5] T. C. M. University, "The "Only" Coke Machine on the Internet," The Carnegie Mellon University, [Online]. Available: https://www.cs.cmu.edu/~coke/history_long.txt. [Accessed 18 March 2019].

[6] E. Dukes, "The Cost of IoT Sensors Is Dropping Fast," iOfficeCorp, 11 September 2018. [Online]. Available: https://www.iofficecorp.com/blog/cost-of-iot-sensors. [Accessed 19 March 2019].

[7] K. L. Lueth, "State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating," IoT Analytics, 8 August 2018. [Online]. Available: https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/. [Accessed 19 March 2019].

[8] Clickatell, "The current state of the Internet of Things," Clickatell, 02 November 2017. [Online]. Available: https://www.clickatell.com/articles/technology/state-internet-of-things/. [Accessed 19 March 2019].

[9] N. Patel, "Internet of things in healthcare: applications, benefits, and challenges," Peerbits, 26 July 2018. [Online]. Available: https://www.peerbits.com/blog/internet-of-things-healthcare-applications-benefits-and-challenges.html. [Accessed 19 March 2019].

[10] M. Kranz, "6 ways the Internet of Things is improving our lives," World Economic Forum, 11 January 2018. [Online]. Available: https://www.weforum.org/agenda/2018/01/6-ways-the-internet-of-things-is-improving-our-lives/. [Accessed 19 March 2019].

[11] P. Suresh, J. V. Daniel, V. Parthasarathy and R. H. Aswathy, "A state of the art review on the Internet of Things (IoT) history, technology and fields of deployment," in *2014 International Conference on Science Engineering and Management Research (ICSEMR)*, Chennai, India, 2015.

[12] F. Insights, "4 Reasons Why Architecting For Change Is Critical For IoT," Forbes, 2017 December 2018. [Online]. Available: https://www.forbes.com/sites/insights-hitachi/2017/12/18/4-reasons-that-architecting-for-change-is-critical-for-iot/#3655ee1a3a0d. [Accessed 20 March 2019].

[13] F. Pereira, R. Correia and N. B. Carvalho, "Passive Sensors for Long Duration Internet of Things Networks," Sensors, 3 October 2017. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5676605/. [Accessed 20 March 2019].

[14] BTI, "6 Leading Types of IoT Wireless Tech and Their Best Use Cases," Behr Technologies Inc., 16 January 2019. [Online]. Available: https://behrtechnologies.com/blog/6-leading-types-of-iot-wireless-tech-and-their-best-use-cases/. [Accessed 21 March 2019].

[15] M. Knight, "Data Management and the Internet of Things," Dataversity, 12 December 2018. [Online]. Available: https://www.dataversity.net/data-management-internet-things/. [Accessed 21 March 2019].

[16] A. Dellinger, "INTERNET OF THINGS SAFETY: NEARLY HALF OF U.S. FIRMS USING IOT HIT BY SECURITY BREACHES," Newsweek, 6 May 2017. [Online]. Available: https://www.newsweek.com/iot-security-internet-things-safety-breaches-businesses-how-protect-621230. [Accessed 21 March 2019].

[17] D. Gates, "Flawed analysis, failed oversight: How Boeing, FAA certified the suspect 737 MAX flight control system," The Seattle Times, 17 March 2019. [Online]. Available: https://www.seattletimes.com/business/boeing-aerospace/failed-certification-faa-missed-safety-issues-in-the-737-max-system-implicated-in-the-lion-air-crash/. [Accessed 2 June 2019].

[18] Satista, "Advertising revenue of Google from 2001 to 2018 (in billion U.S. dollars)," Satista, February 2019. [Online]. Available: https://www.statista.com/statistics/266249/advertising-revenue-of-google/. [Accessed 5 March 2019].

[19] Satista, "Annual revenue of Google from 2002 to 2018 (in billion U.S. dollars)," Satista, February 2019. [Online]. Available: https://www.statista.com/statistics/266206/googles-annual-global-revenue/. [Accessed 5 March 2019].

[20] L. Stahl, "Aleksandr Kogan: The link between Cambridge Analytica and Facebook," CBS News, 22 April 2018. [Online]. Available: https://www.cbsnews.com/news/aleksandr-kogan-the-link-between-cambridge-analytica-and-facebook/. [Accessed 7 March 2019].

[21] J. C. Wong, "Mark Zuckerberg faces tough questions in two-day congressional testimony – as it happened," The Guardian, 11 April 2018. [Online]. Available: https://www.theguardian.com/technology/live/2018/apr/11/mark-zuckerberg-testimony-live-updates-house-congress-cambridge-analytica?page=with:block-5ace2921e4b08f6cf5be55e4#block-5ace2921e4b08f6cf5be55e4. [Accessed 4 March 2019].

[22] D. MacMillan, "Tech's 'Dirty Secret': The App Developers Sifting Through Your Gmail," The Wall Street Journal, 2 July 2018. [Online]. Available: https://www.wsj.com/articles/techs-dirty-secret-the-app-developers-sifting-through-your-gmail-1530544442. [Accessed 5 March 2019].

[23] J. D. McKinnon and D. MacMillan, "Google Says It Continues to Allow Apps to Scan Data From Gmail Accounts," The Wall Street Journal, 20 September 2018. [Online]. Available: https://www.wsj.com/articles/google-says-it-continues-to-allow-apps-to-scan-data-from-gmail-accounts-1537459989. [Accessed 5 March 2019].

[24] J. Emont, L. Stevens and R. MacMillan, "Amazon Investigates Employees Leaking Data for Bribes," The Wall Street Journal, 16 September 2018. [Online]. Available: https://www.wsj.com/articles/amazon-investigates-employees-leaking-data-for-bribes-1537106401. [Accessed 5 March 2019].

[25] M. Bellis, "The History of the Thermometer," ThoughtCo, 27 November 2018. [Online]. Available: https://www.thoughtco.com/the-history-of-the-thermometer-1992525. [Accessed 6 March 2019].

[26] D. G. M. Jr., "'Smart Thermometers' Track Flu Season in Real Time," The New York Times, 16 January 2018. [Online]. Available: https://www.nytimes.com/2018/01/16/health/smart-thermometers-flu.html?module=inline. [Accessed 5 March 2019].

[27] S. Maheshwari, "This Thermometer Tells Your Temperature, Then Tells Firms Where to Advertise," The New York Times, 23 October 2018. [Online]. Available: https://www.nytimes.com/2018/10/23/business/media/fever-advertisements-medicine-clorox.html. [Accessed 5 March 2019].

[28] L. Constantin, "How to secure your router and home network," PCWorld, 08 July 2016. [Online]. Available: https://www.pcworld.com/article/3093362/how-to-secure-your-router-and-home-network.html. [Accessed 22 March 2019].

[29] M. Rouse, "Unified threat management devices: Understanding UTM and its vendors," TechTarget, March 2018. [Online]. Available: https://searchsecurity.techtarget.com/definition/unified-threat-management-UTM. [Accessed 22 March 2019].

[30] Luma, "LumaGuardian," Luma, [Online]. Available: https://lumahome.com/. [Accessed 24 March 2019].

[31] "Bitdefender BOX," Bitdefender, [Online]. Available: https://www.bitdefender.com/box/. [Accessed 24 March 2019].

[32] "Verizon Home Network Protection," Verizon, [Online]. Available: https://www.verizon.com/support/residential/internet/essentials/home-network-protection. [Accessed 24 March 2019].

[33] E. Networks, "Extreme Defender for IoT," Extreme Networks, 2019. [Online]. Available: https://www.extremenetworks.com/product/extreme-defender-for-iot/. [Accessed 24 March 2019].

[34] F-Secure, "SENSE security router," F-Secure, 2019. [Online]. Available: https://www.f-secure.com/en/web/home_global/sense. [Accessed 24 March 2019].

[35] "Securing the Internet of Things," Bullguard, [Online]. Available: https://dojo.bullguard.com/. [Accessed 24 March 2019].

[36] C. AI, "Connected Experience Built with Trust," CUJO LLC, [Online]. Available: https://www.getcujo.com/. [Accessed 24 March 2019].

[37] B. Dickson, "4 devices that can help secure your home's IoT," Insider, 4 January 2016. [Online]. Available: https://thenextweb.com/insider/2016/01/04/4-devices-that-can-help-secure-your-homes-iot/. [Accessed 24 March 2019].

[38] P. Bull, R. Austin, E. Popov, M. Sharma and R. Watson, "Flow Based Security for IoT Devices Using an SDN Gateway," in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, Vienna, Austria, 2016.

[39] J. Folkens, "Building a gateway to the Internet of Things," Texas Instruments, December 2014. [Online]. Available: http://www.ti.com/lit/wp/spmy013/spmy013.pdf. [Accessed 30 June 2019].

[40] J. King and A. I. Awad, "A Distributed Security Mechanism for Resource-Constrained IoT Devices," Informatica, 30 October 2015. [Online]. Available: http://www.informatica.si/index.php/informatica/article/view/1046/841. [Accessed 30 June 2019].

[41] E. A. Posner and E. G. Weyl, "Data as Labor - Radical Markets," Radical Markets, 3 September 2017. [Online]. Available: http://radicalmarkets.com/chapters/data-as-labor/. [Accessed 25 March 2019].

[42] I. A. Ibarra, L. Goff, D. J. Hernández, J. Lanier and E. G. Weyl, "Should we treat data as labor? Let's open up the discussion," Brookings, 21 February 2018. [Online]. Available: https://www.brookings.edu/blog/techtank/2018/02/21/should-we-treat-data-as-labor-lets-open-up-the-discussion/. [Accessed 25 March 2019].

[43] "The world's most valuable resource is no longer oil, but data," The Economist Group Limited, 6 May 2017. [Online]. Available: https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data. [Accessed 25 March 2019].

[44] "Data workers of the world, unite," The Economist Group Limited, 7 July 2018. [Online]. Available: https://www.economist.com/the-world-if/2018/07/07/data-workers-of-the-world-unite. [Accessed 11 March 2019].

[45] S. Öman, "Implementing Data Protection in Law," Stockholm Institute for Scandianvian Law 1957-2010.

[46] "PIPEDA fair information principles," Office of the Privacy Commissioner of Canada, [Online]. Available: https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/p_principle/. [Accessed 27 March 2019].

[47] "General Data Protection Regulation GDPR," Intersoft Consulting, [Online]. Available: https://gdpr-info.eu/. [Accessed 26 March 2019].

[48] J. J. Roberts, "The GDPR Is in Effect: Should U.S. Companies Be Afraid?," Fortune, 25 May 2018. [Online]. Available: http://fortune.com/2018/05/24/the-gdpr-is-in-effect-should-u-s-companies-be-afraid/. [Accessed 30 March 2019].

[49] PIPEDA, "Joint investigation of Facebook, Inc. by the Privacy Commissioner of Canada and the Information and Privacy Commissioner for British Columbia," Office of the Privacy Commisionner of Canada, 25 April 2019. [Online]. Available: https://www.priv.gc.ca/en/opc-actions-and-decisions/investigations/investigations-into-businesses/2019/pipeda-2019-002/. [Accessed 8 June 2019].

[50] "Summary of the HIPAA Privacy Rule," U.S. Department of Health & Human Services, [Online]. Available: https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/index.html. [Accessed 27 March 2019].

[51] "Fair and Accurate Credit Transactions Act of 2003," Federal Trade Commission, [Online]. Available: https://www.ftc.gov/enforcement/statutes/fair-accurate-credit-transactions-act-2003. [Accessed 27 March 2019].

[52] "PART 682—DISPOSAL OF CONSUMER REPORT INFORMATION AND RECORDS," Electronic Code of Federal Regulations, [Online]. Available: https://www.ecfr.gov/cgi-bin/text-

idx?c=ecfr&SID=636cbcb63dd25d85afab80ce0f5817ed&rgn=div5&view=text&node=16%3A1.0.1.6.80&idno=16. [Accessed 27 March 2019].

[53] T. Cook, "You Deserve Privacy Online. Here's How You Could Actually Get It," Time, 12 September 2018. [Online]. Available: http://time.com/collection/davos-2019/5502591/tim-cook-data-privacy/. [Accessed 27 March 2019].

[54] M. Rouse, "distributed ledger technology (DLT)," TechTarget, August 2017. [Online]. Available: https://searchcio.techtarget.com/definition/distributed-ledger. [Accessed 19 May 2019].

[55] L. Fortney, "Blockchain, Explained," Investopedia, 10 February 2019. [Online]. Available: https://www.investopedia.com/terms/b/blockchain.asp. [Accessed 19 May 2019].

[56] B. Asolo, "IOTA Explained," Mycryptopedia, 1 November 2018. [Online]. Available: https://www.mycryptopedia.com/iota-explained/. [Accessed 19 May 2019].

[57] A. Davies, "IoT & Blockchain Technology: Uses Cases Overview," DevTeam.Space, 15 March 2019. [Online]. Available: https://www.devteam.space/blog/iot-blockchain-technology-uses-cases-overview/. [Accessed 30 March 2019].

[58] N. Kshetri, "Can Blockchain Strengthen the Internet of Things?," IEEE, 17 August 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8012302. [Accessed 30 March 2019].

[59] "ethereum Blockchain App Platform," Ethereum, [Online]. Available: https://www.ethereum.org/. [Accessed 30 March 2019].

[60] S. Huh, S. Cho and S. Kim, "Managing IoT devices using blockchain platform," in *2017 19th International Conference on Advanced Communication Technology (ICACT)*, Bongpyeong, South Korea, 2017.

[61] Bitstamp, "Should I run my own node?," Medium, 15 June 2018. [Online]. Available: https://medium.com/bitstamp-blog/should-i-run-my-own-node-13c3f6a21627. [Accessed 31 March 2019].

[62] I. Foundation, "What is IOTA?," IOTA Foundation, [Online]. Available: https://www.iota.org/get-started/what-is-iota. [Accessed 31 March 2019].

[63] N. Gehrke, "IOTA: The Misunderstood Coin?," Medium, 26 December 2017. [Online]. Available: https://medium.com/@norbert.gehrke/iota-the-misunderstood-coin-c6c94678ec99. [Accessed 31 March 2019].

[64] D. Hundeyin, "IOTA Outlines Plans for Killing Off its Centralized 'Coordinator'," CCN, 23 November 2018. [Online]. Available: https://www.ccn.com/iota-outlines-plans-for-killing-off-its-centralized-coordinator. [Accessed 31 March 2019].

[65] K. Sir, "Charles Liu: Smart Contracts in DAG," Medium, 29 October 2018. [Online]. Available: https://medium.com/vitelabs/smart-contracts-in-dag-5059250b916b. [Accessed 31 March 2019].

[66] "VITE," VITE Labs, [Online]. Available: http://www.vite.org/. [Accessed 31 March 2019].

[67] "Qubic," Medium, 03 June 2018. [Online]. Available: https://qubic.iota.org/. [Accessed 31 March 2019].

[68] "5 advantages (and 1 disadvantage) of zero knowledge authentication," Ingram Micro, 12 November 2018. [Online]. Available: https://imaginenext.ingrammicro.com/trends/november-2018/5-advantages-and-1-disadvantage-of-zero-knowledge-authentication. [Accessed 1 April 2019].

[69] N. Mancall-Bitel, "The 'smart' baby technology raising today's children," BBC, 29 November 2018. [Online]. Available: http://www.bbc.com/capital/story/20181128-the-smart-baby-technology-raising-todays-children. [Accessed 5 March 2019].

[70] N. Shields, "New survey shows consumers are wary of smart home devices invading their privacy," Business Insider, 26 April 2018. [Online]. Available: https://www.businessinsider.com/survey-says-consumers-have-privacy-concerns-with-smart-home-devices-2018-4. [Accessed 3 April 2019].

[71] N. Feamster, N. Apthorpe, D. Y. Huang, G. Acar, F. Li and A. Narayanan, "Announcing IoT Inspector: Studying Smart Home IoT Device Behavior," Freedom To Tinker, 23 April 2018. [Online]. Available: https://freedom-to-tinker.com/2018/04/23/announcing-iot-inspector-a-tool-to-study-smart-home-iot-device-behavior/. [Accessed 5 March 2019].

[72] Internet Association, [Online]. Available: https://internetassociation.org/. [Accessed 7 March 2019].

[73] Industrial Internet Consortium, [Online]. Available: https://www.iiconsortium.org/. [Accessed 7 March 2019].

[74] J. Doward and A. Gibbs, "Did Cambridge Analytica influence the Brexit vote and the US election?," The Guardian, 4 March 2017. [Online]. Available: https://www.theguardian.com/politics/2017/mar/04/nigel-oakes-cambridge-analytica-what-role-brexit-trump. [Accessed 5 March 2019].

[75] N. Confessore, "Cambridge Analytica and Facebook: The Scandal and the Fallout So Far," The New York Times, 4 April 2018. [Online]. Available: https://www.nytimes.com/2018/04/04/us/politics/cambridge-analytica-scandal-fallout.html. [Accessed 5 March 2019].

[76] E. P. I. Center, "Internet of Things (IoT)," Electronic Privacy Information Center, [Online]. Available: https://epic.org/privacy/internet/iot/. [Accessed 5 March 2019].

[77] N. Gibbs and L. Grossman, "Here's the Full Transcript of TIME's Interview With Apple CEO Tim Cook," Time, 17 March 2016. [Online]. Available: http://time.com/4261796/tim-cook-transcript/. [Accessed 5 March 2019].

[78] M. U. IOTAP, "IoT, Security and Privacy," Medium, 14 June 2016. [Online]. Available: https://medium.com/@iotap/internet-of-things-security-and-privacy-78bc0a41881b. [Accessed 6 March 2019].

[79] M. U. IOTAP, "On Privacy and Security in Smart Homes," Medium, 14 June 2016. [Online]. Available: https://medium.com/@iotap/on-privacy-and-security-in-smart-homes-543f62aa9917. [Accessed 6 March 2019].

[80] C. Kolias, G. Kambourakis, A. Stavrou and J. Voas, "DDoS in the IoT: Mirai and Other Botnets," IEEE, 7 July 2017. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7971869. [Accessed 10 March 2019].

[81] "The search engine for the Internet of Things," Shodan, [Online]. Available: https://www.shodan.io/. [Accessed 02 May 2019].

[82] "Raspberry Pi 3 Model B+," Raspberry Pi Foundation, [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. [Accessed 02 May 2019].

[83] "What Is DD-WRT Firmware?," Flash Routers, [Online]. Available: https://www.flashrouters.com/learn/router-basics/what-is-dd-wrt. [Accessed 13 April 2019].

[84] "Snort," Cisco, 2019. [Online]. Available: https://www.snort.org/. [Accessed 13 April 2019].

[85] N. B. Beser, "Operating cable modems in a low power mode". U.S. Patent 7,389,528, 17 June 2008.

[86] A. Schwartzman and C. Leano, "Methods and apparatus for enabling and disabling cable modem receiver circuitry". U.S. Patent 7,587,746, 8 September 2009.

[87] J. Frankenfield, "Permissioned Blockchains," Investopedia, 10 April 2018. [Online]. Available: https://www.investopedia.com/terms/p/permissioned-blockchains.asp. [Accessed 1 April 2019].

[88] "Introduction to Akka," Akka, [Online]. Available: https://doc.akka.io/docs/akka/2.5.3/scala/guide/introduction.html. [Accessed 29 April 2019].

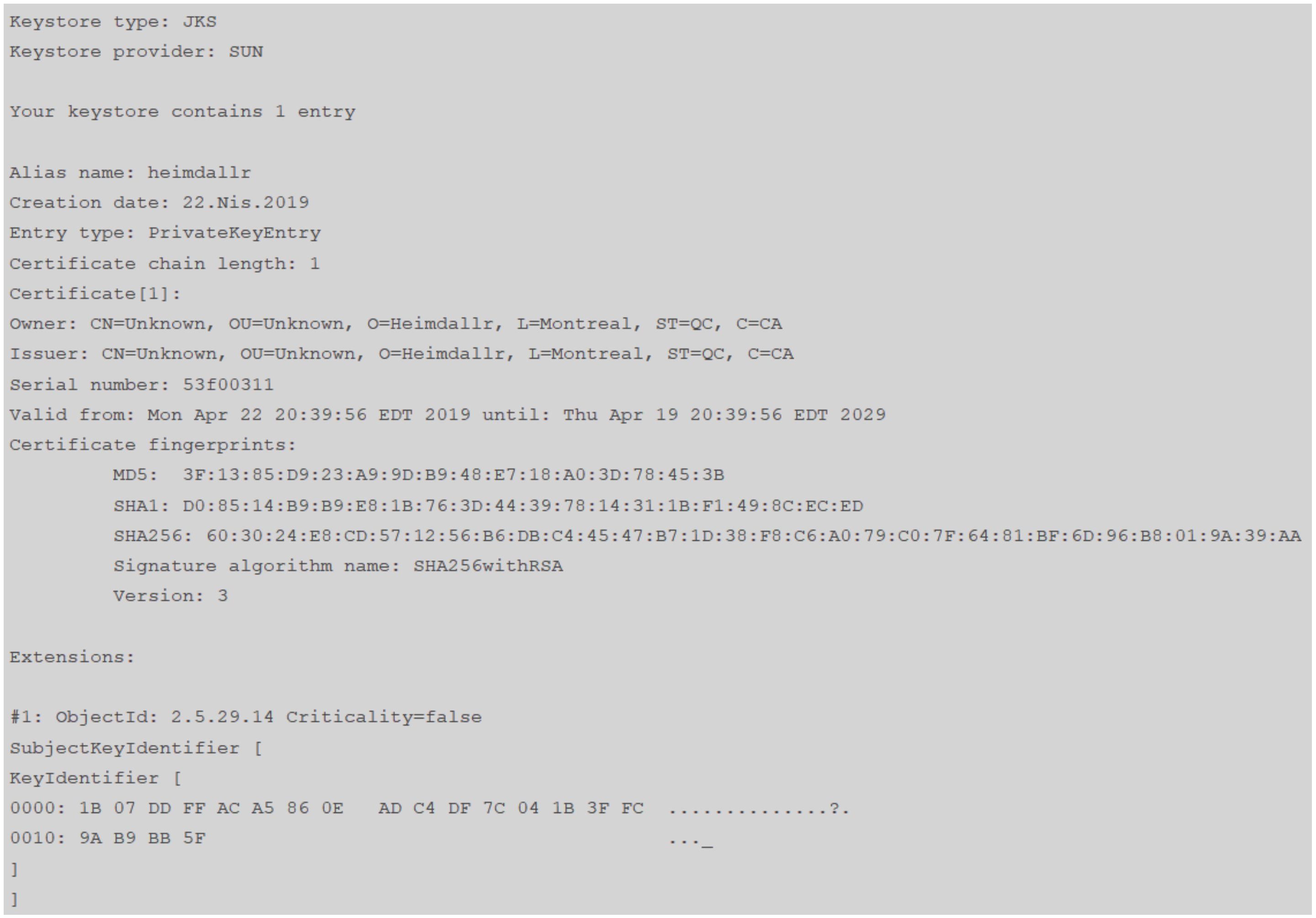
[89] X. Jia, Q. Feng, T. Fan and Q. Lei, "RFID technology and its applications in Internet of Things (IoT)," in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, Yichang, China, 2012.

[90] L. Tian, J. Famaey and S. Latré, "Evaluation of the IEEE 802.11ah Restricted Access Window mechanism for dense IoT networks," in *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Coimbra, Portugal, 2016.

8 Appendices

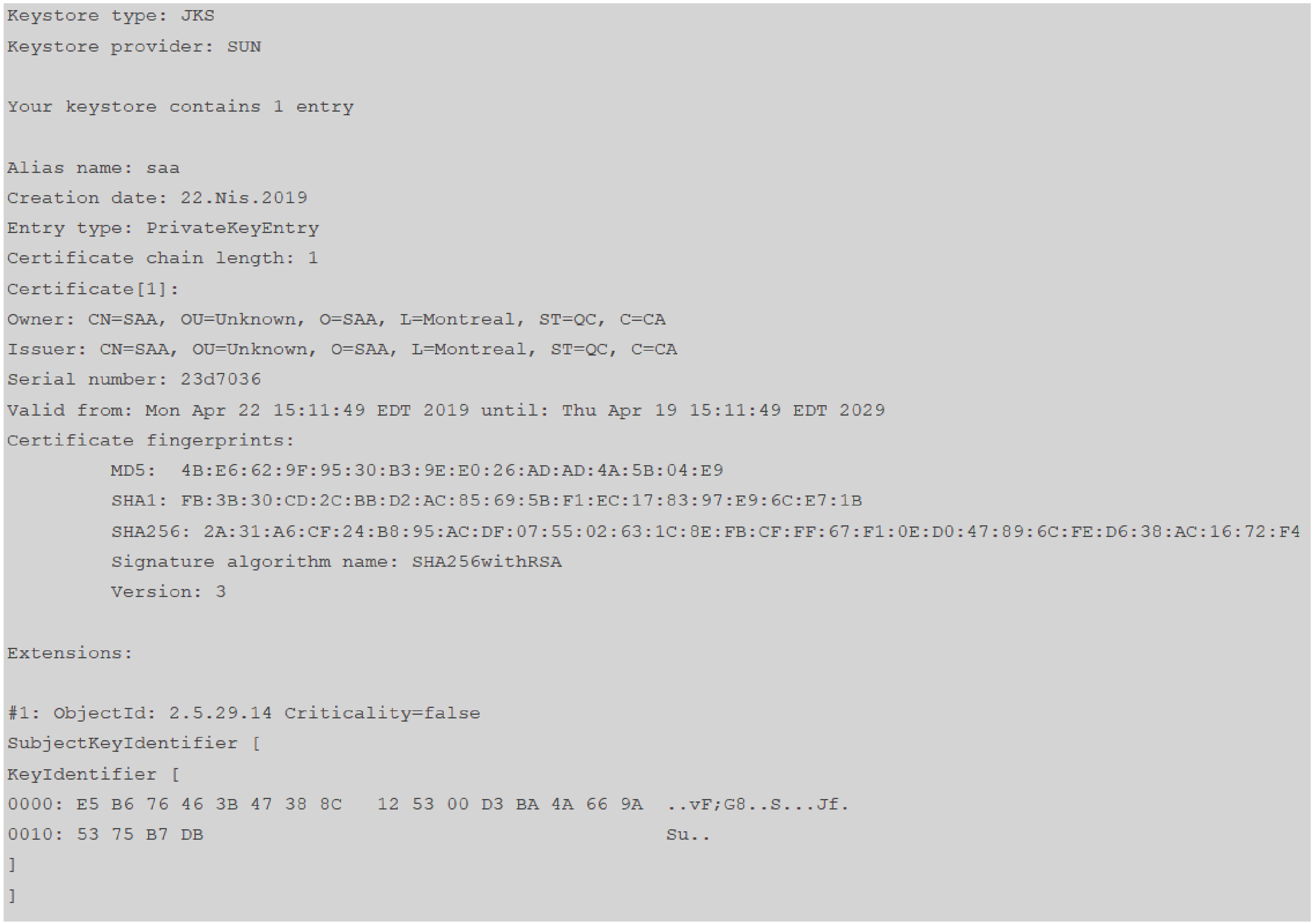
- **Appendix A** is proving the figures that are showing the details of self-signed TLS certificates of Heimdallr and SAA.
- **Appendix B** is providing total of 6 figures which are the zoomed in parts of the intuitive flow of the Heimdallr system diagram given in Figure 15.
- **Appendix C** is providing the screenshot of the VisualVM, the software monitoring tool that we used to analyze Heimdallr software, during our experimentation.

8.1 Appendix A



```
Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: heimdallr
Creation date: 22.Nis.2019
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Unknown, OU=Unknown, O=Heimdallr, L=Montreal, ST=QC, C=CA
Issuer: CN=Unknown, OU=Unknown, O=Heimdallr, L=Montreal, ST=QC, C=CA
Serial number: 53f00311
Valid from: Mon Apr 22 20:39:56 EDT 2019 until: Thu Apr 19 20:39:56 EDT 2029
Certificate fingerprints:
         MD5:  3F:13:85:D9:23:A9:9D:B9:48:E7:18:A0:3D:78:45:3B
         SHA1: D0:85:14:B9:B9:E8:1B:76:3D:44:39:78:14:31:1B:F1:49:8C:EC:ED
         SHA256: 60:30:24:E8:CD:57:12:56:B6:DB:C4:45:47:B7:1D:38:F8:C6:A0:79:C0:7F:64:81:BF:6D:96:B8:01:9A:39:AA
         Signature algorithm name: SHA256withRSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 1B 07 DD FF AC A5 86 0E   AD C4 DF 7C 04 1B 3F FC  ..............?.
0010: 9A B9 BB 5F                                        ..._
]
]
```

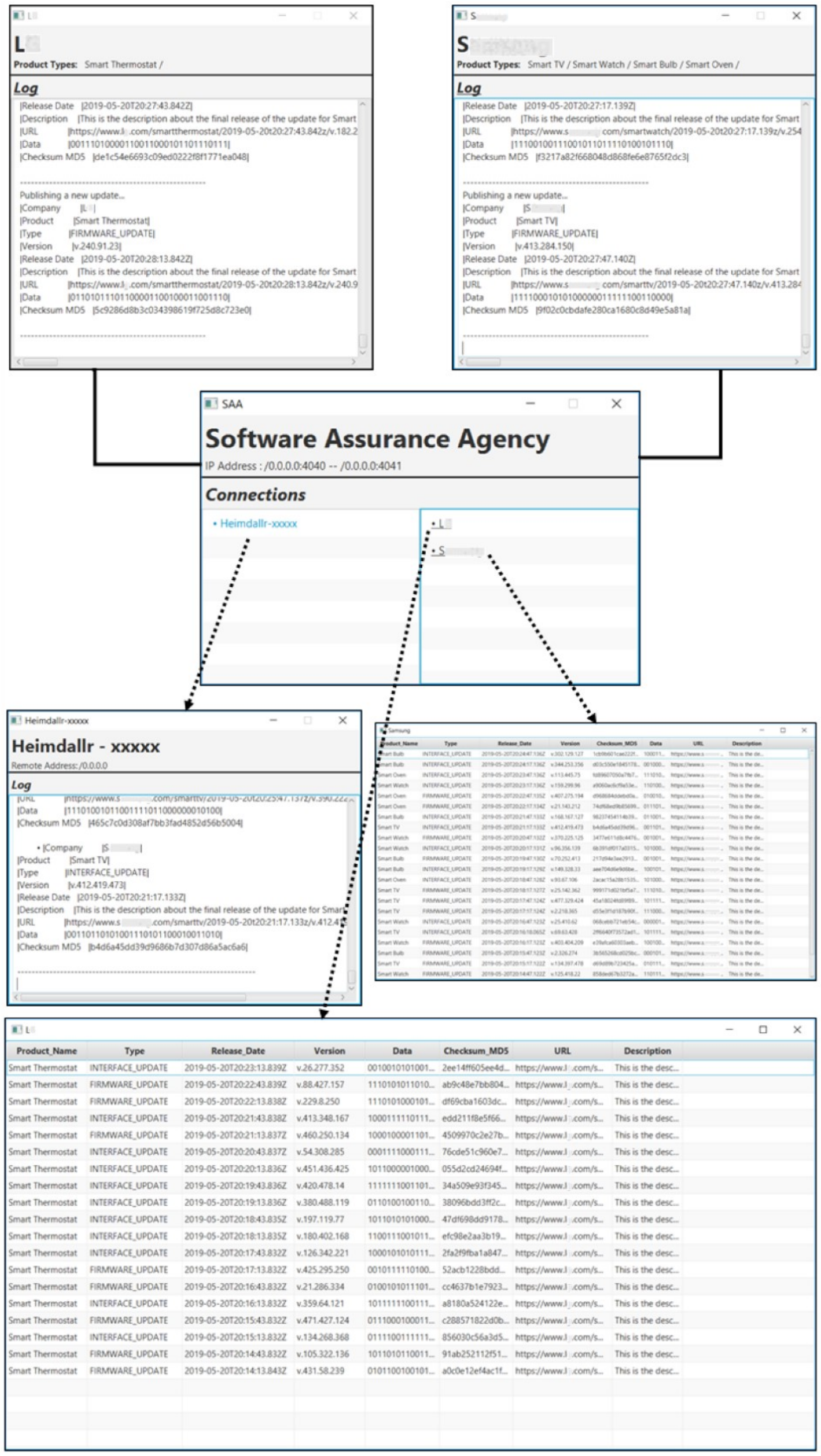
Figure 16: Heimdallr's Self-Signed TLS Certificate
This figure shows Heimdallr's self-signed TLS certificate to be used in secure socket communication over TCP



```
Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry


Alias name: saa
Creation date: 22.Nis.2019
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=SAA, OU=Unknown, O=SAA, L=Montreal, ST=QC, C=CA
Issuer: CN=SAA, OU=Unknown, O=SAA, L=Montreal, ST=QC, C=CA
Serial number: 23d7036
Valid from: Mon Apr 22 15:11:49 EDT 2019 until: Thu Apr 19 15:11:49 EDT 2029
Certificate fingerprints:
         MD5:  4B:E6:62:9F:95:30:B3:9E:E0:26:AD:AD:4A:5B:04:E9
         SHA1: FB:3B:30:CD:2C:BB:D2:AC:85:69:5B:F1:EC:17:83:97:E9:6C:E7:1B
         SHA256: 2A:31:A6:CF:24:B8:95:AC:DF:07:55:02:63:1C:8E:FB:CF:FF:67:F1:0E:D0:47:89:6C:FE:D6:38:AC:16:72:F4
         Signature algorithm name: SHA256withRSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: E5 B6 76 46 3B 47 38 8C   12 53 00 D3 BA 4A 66 9A  ..vF;G8..S...Jf.
0010: 53 75 B7 DB                                        Su..
]
]
```

Figure 17: SAA's Self-Signed TLS Certificate
This figure shows SAA's self-signed TLS certificate to be used in secure socket communication over TCP

8.2 Appendix B



L

Product Types: Smart Thermostat /

Log

```
|Release Date   |2019-05-20T20:27:43.842Z|
|Description    |This is the description about the final release of the update for Smart
|URL            |https://www.l .com/smartthermostat/2019-05-20t20:27:43.842z/v.182.2
|Data           |00111010000110011000101101110111|
|Checksum MD5   |de1c54e6693c09ed0222f8f1771ea048|

--------------------------------------------------
Publishing a new update...
|Company        |L |
|Product        |Smart Thermostat|
|Type           |FIRMWARE_UPDATE|
|Version        |v.240.91.23|
|Release Date   |2019-05-20T20:28:13.842Z|
|Description    |This is the description about the final release of the update for Smart
|URL            |https://www.l .com/smartthermostat/2019-05-20t20:28:13.842z/v.240.9
|Data           |01101011101100001100100011001110|
|Checksum MD5   |5c9286d8b3c034398619f725d8c723e0|

--------------------------------------------------
```

S

Product Types: Smart TV / Smart Watch / Smart Bulb / Smart Oven /

Log

```
|Release Date   |2019-05-20T20:27:17.139Z|
|Description    |This is the description about the final release of the update for Smart
|URL            |https://www.s com/smartwatch/2019-05-20t20:27:17.139z/v.254
|Data           |11100100111001011011110100101110|
|Checksum MD5   |f3217a82f668048d868fe6e8765f2dc3|

--------------------------------------------------
Publishing a new update...
|Company        |S |
|Product        |Smart TV|
|Type           |FIRMWARE_UPDATE|
|Version        |v.413.284.150|
|Release Date   |2019-05-20T20:27:47.140Z|
|Description    |This is the description about the final release of the update for Smart
|URL            |https://www.s com/smarttv/2019-05-20t20:27:47.140z/v.413.284
|Data           |11110001010100000011111100110000|
|Checksum MD5   |9f02c0cbdafe280ca1680c8d49e5a81a|

--------------------------------------------------
```

SAA

Software Assurance Agency

IP Address : /0.0.0.0:4040 -- /0.0.0.0:4041

Connections

- Heimdallr-xxxxx

- L
- S

Heimdallr-xxxxx

Heimdallr - xxxxx

Remote Address: /0.0.0.0

Log

```
|URL            |https://www.s .com/smarttv/2019-05-20t20:25:47.137z/v.390.222
|Data           |11101001011001111011000000010100|
|Checksum MD5   |465c7c0d308af7bb3fad4852d56b5004|

      • |Company        |S |
|Product        |Smart TV|
|Type           |INTERFACE_UPDATE|
|Version        |v.412.419.473|
|Release Date   |2019-05-20T20:21:17.133Z|
|Description    |This is the description about the final release of the update for Smart
|URL            |https://www.s .com/smarttv/2019-05-20t20:21:17.133z/v.412.41
|Data           |00110110101001110101100010011010|
|Checksum MD5   |b4d6a45dd39d9686b7d307d86a5ac6a6|

------------------------------------------------------------
```

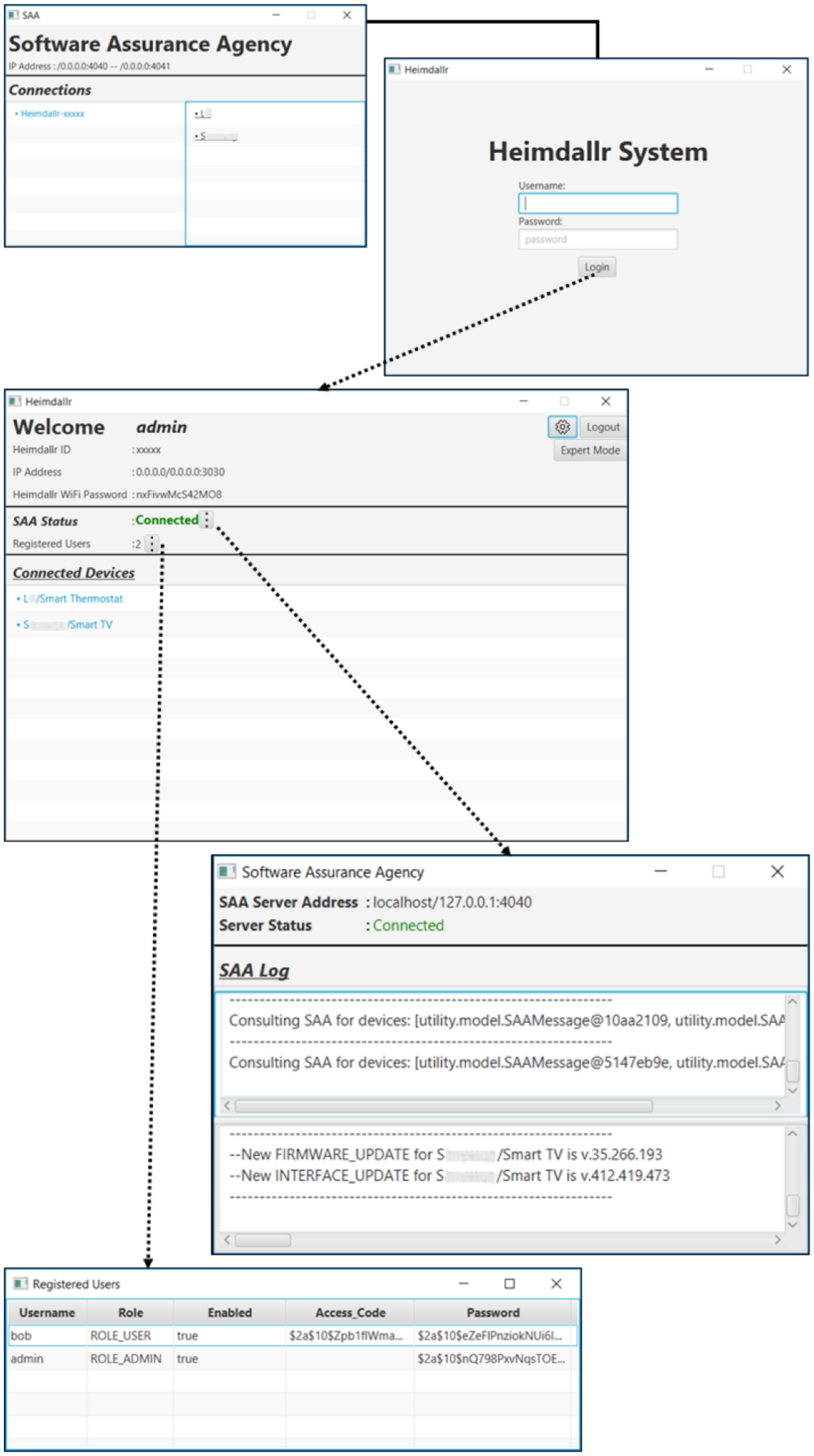
Samsung

Product_Name	Type	Release_Date	Version	Checksum_MD5	Data	URL	Description
Smart Bulb	INTERFACE_UPDATE	2019-05-20T20:24:47.136Z	v.302.129.127	1cb9b601cae222f...	100011...	https://www.s...	This is the de...
Smart Bulb	INTERFACE_UPDATE	2019-05-20T20:24:17.136Z	v.344.253.356	d03c550e1845178...	001000...	https://www.s...	This is the de...
Smart Oven	INTERFACE_UPDATE	2019-05-20T20:23:47.136Z	v.113.445.75	fd89607050a7fb7...	111010...	https://www.s...	This is the de...
Smart Watch	INTERFACE_UPDATE	2019-05-20T20:23:17.136Z	v.159.299.96	a9060ac6cf9a53e...	110100...	https://www.s...	This is the de...
Smart Oven	FIRMWARE_UPDATE	2019-05-20T20:22:47.135Z	v.407.275.194	d968684ddebd0a...	010010...	https://www.s...	This is the de...
Smart Oven	FIRMWARE_UPDATE	2019-05-20T20:22:17.134Z	v.21.140.212	74df68ed9b85699...	011101...	https://www.s...	This is the de...
Smart Bulb	INTERFACE_UPDATE	2019-05-20T20:21:47.133Z	v.168.167.127	98237454114b39...	011001...	https://www.s...	This is the de...
Smart TV	INTERFACE_UPDATE	2019-05-20T20:21:17.133Z	v.412.419.473	b4d6a45dd39d96...	001101...	https://www.s...	This is the de...
Smart Watch	FIRMWARE_UPDATE	2019-05-20T20:20:47.132Z	v.370.225.125	3477e611d8c4476...	001001...	https://www.s...	This is the de...
Smart Watch	INTERFACE_UPDATE	2019-05-20T20:20:17.131Z	v.96.356.139	6b391df017a0315...	101000...	https://www.s...	This is the de...
Smart Bulb	FIRMWARE_UPDATE	2019-05-20T20:19:47.130Z	v.70.252.413	217d94e3ee2913...	001001...	https://www.s...	This is the de...
Smart Bulb	INTERFACE_UPDATE	2019-05-20T20:19:17.129Z	v.149.328.33	aee704d6e9d6be...	100101...	https://www.s...	This is the de...
Smart Oven	INTERFACE_UPDATE	2019-05-20T20:18:47.128Z	v.93.67.106	2acac15a28b1535...	101000...	https://www.s...	This is the de...
Smart TV	FIRMWARE_UPDATE	2019-05-20T20:18:17.127Z	v.25.142.362	999171d021bf5a7...	111010...	https://www.s...	This is the de...
Smart TV	FIRMWARE_UPDATE	2019-05-20T20:17:47.124Z	v.477.329.424	45a18024fd89f89...	101111...	https://www.s...	This is the de...
Smart TV	FIRMWARE_UPDATE	2019-05-20T20:17:17.124Z	v.2.218.365	d55e3f1d187b90f...	111000...	https://www.s...	This is the de...
Smart Watch	INTERFACE_UPDATE	2019-05-20T20:16:47.123Z	v.25.410.62	068cebb721eb54c...	000001...	https://www.s...	This is the de...
Smart TV	INTERFACE_UPDATE	2019-05-20T20:16:18.065Z	v.69.63.428	2ff6640f73572ad1...	101111...	https://www.s...	This is the de...
Smart Watch	FIRMWARE_UPDATE	2019-05-20T20:16:17.123Z	v.403.404.209	e39afca60303aeb...	100100...	https://www.s...	This is the de...
Smart Bulb	FIRMWARE_UPDATE	2019-05-20T20:15:47.123Z	v.2.326.274	3b565268cd025bc...	000101...	https://www.s...	This is the de...
Smart TV	FIRMWARE_UPDATE	2019-05-20T20:15:17.122Z	v.134.397.478	d69d89b723425a...	010111...	https://www.s...	This is the de...
Smart Watch	FIRMWARE_UPDATE	2019-05-20T20:14:47.122Z	v.125.418.22	858ded67b3272a...	110111...	https://www.s...	This is the de...

L

Product_Name	Type	Release_Date	Version	Data	Checksum_MD5	URL	Description
Smart Thermostat	INTERFACE_UPDATE	2019-05-20T20:23:13.839Z	v.26.277.352	0010010101001...	2ee14ff605ee4d...	https://www.l .com/s...	This is the desc...
Smart Thermostat	FIRMWARE_UPDATE	2019-05-20T20:22:43.839Z	v.88.427.157	1110101011010...	ab9c48e7bb804...	https://www.l .com/s...	This is the desc...
Smart Thermostat	FIRMWARE_UPDATE	2019-05-20T20:22:13.838Z	v.229.8.250	1110101000101...	df69cba1603dc...	https://www.l .com/s...	This is the desc...
Smart Thermostat	INTERFACE_UPDATE	2019-05-20T20:21:43.838Z	v.413.348.167	1000111110111...	edd211f8e5f66...	https://www.l .com/s...	This is the desc...
Smart Thermostat	FIRMWARE_UPDATE	2019-05-20T20:21:13.837Z	v.460.250.134	1000100001101...	4509970c2e27b...	https://www.l .com/s...	This is the desc...
Smart Thermostat	INTERFACE_UPDATE	2019-05-20T20:20:43.837Z	v.54.308.285	0001111000111...	76cde51c960e7...	https://www.l .com/s...	This is the desc...
Smart Thermostat	INTERFACE_UPDATE	2019-05-20T20:20:13.836Z	v.451.436.425	1011000001000...	055d2cd24694f...	https://www.l .com/s...	This is the desc...
Smart Thermostat	INTERFACE_UPDATE	2019-05-20T20:19:43.836Z	v.420.478.14	1111111001101...	34a509e93f345...	https://www.l .com/s...	This is the desc...
Smart Thermostat	INTERFACE_UPDATE	2019-05-20T20:19:13.836Z	v.380.488.119	0110100100110...	38096bdd3ff2c...	https://www.l .com/s...	This is the desc...
Smart Thermostat	INTERFACE_UPDATE	2019-05-20T20:18:43.835Z	v.197.119.77	1011010101000...	47df698dd9178...	https://www.l .com/s...	This is the desc...
Smart Thermostat	INTERFACE_UPDATE	2019-05-20T20:18:13.835Z	v.180.402.168	1100111001011...	efc98e2aa3b19...	https://www.l .com/s...	This is the desc...
Smart Thermostat	INTERFACE_UPDATE	2019-05-20T20:17:43.832Z	v.126.342.221	1000101010111...	2fa2f9fba1a847...	https://www.l .com/s...	This is the desc...
Smart Thermostat	FIRMWARE_UPDATE	2019-05-20T20:17:13.832Z	v.425.295.250	0010111110100...	52acb1228bdd...	https://www.l .com/s...	This is the desc...
Smart Thermostat	FIRMWARE_UPDATE	2019-05-20T20:16:43.832Z	v.21.286.334	0100101011101...	cc4637b1e7923...	https://www.l .com/s...	This is the desc...
Smart Thermostat	INTERFACE_UPDATE	2019-05-20T20:16:13.832Z	v.359.64.121	1011111100111...	a8180a524122e...	https://www.l .com/s...	This is the desc...
Smart Thermostat	FIRMWARE_UPDATE	2019-05-20T20:15:43.832Z	v.471.427.124	0111000100011...	c288571822d0b...	https://www.l .com/s...	This is the desc...
Smart Thermostat	INTERFACE_UPDATE	2019-05-20T20:15:13.832Z	v.134.268.368	0111100111111...	856030c56a3d5...	https://www.l .com/s...	This is the desc...
Smart Thermostat	FIRMWARE_UPDATE	2019-05-20T20:14:43.832Z	v.105.322.136	1011010110011...	91ab252112f51...	https://www.l .com/s...	This is the desc...
Smart Thermostat	FIRMWARE_UPDATE	2019-05-20T20:14:13.843Z	v.431.58.239	0101100100101...	a0c0e12ef4ac1f...	https://www.l .com/s...	This is the desc...

Figure 18: Intuitive Flow Diagram of the Heimdallr System (part 1)
This figure gives a closer look at the relationship between the SAA and the two simulated IoT device makers used in Heimdallr system simulation

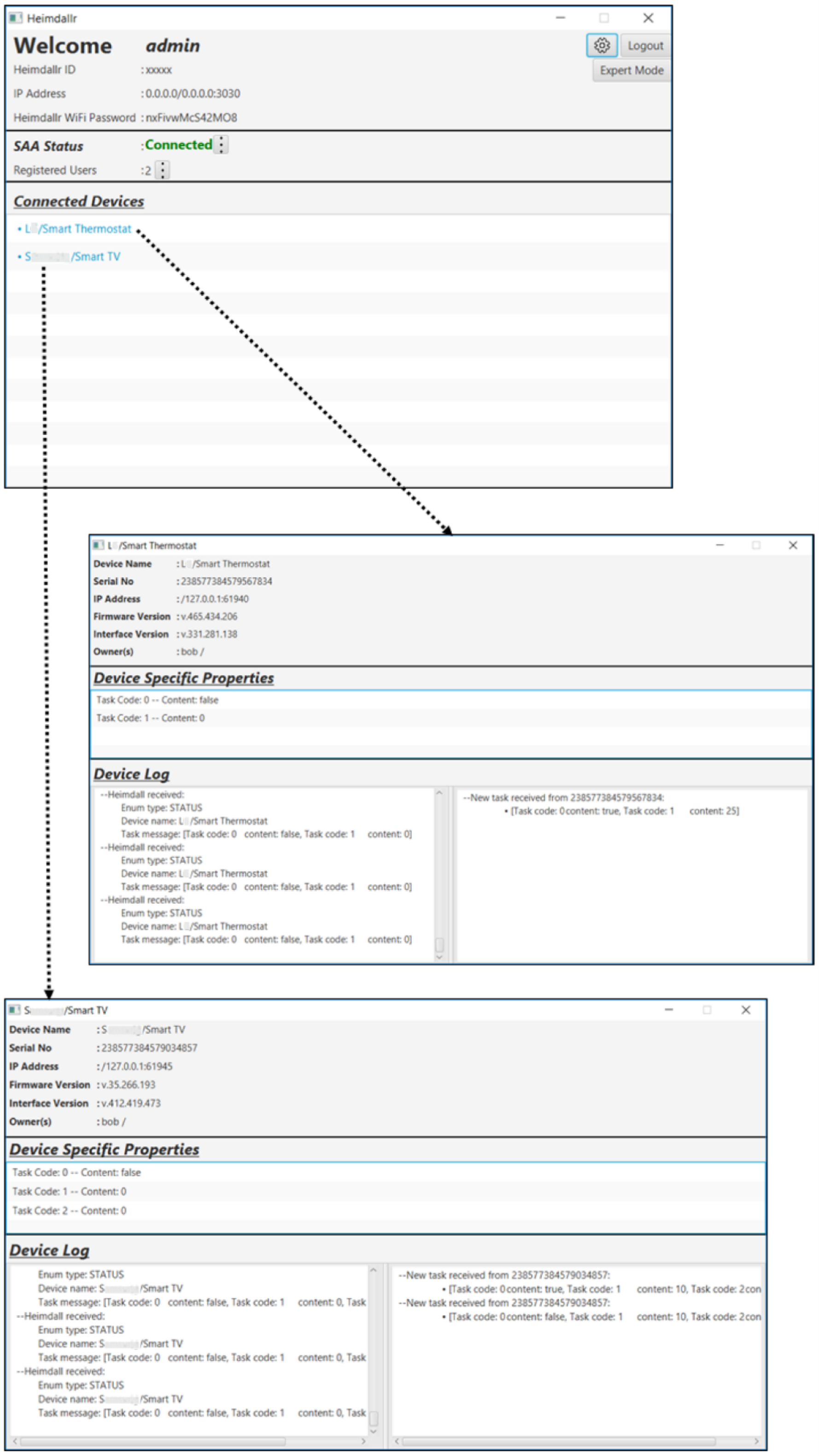


```
-----------------------------------------------------------
Consulting SAA for devices: [utility.model.SAAMessage@10aa2109, utility.model.SAA
-----------------------------------------------------------
Consulting SAA for devices: [utility.model.SAAMessage@5147eb9e, utility.model.SAA
```

```
-----------------------------------------------------------
--New FIRMWARE_UPDATE for S /Smart TV is v.35.266.193
--New INTERFACE_UPDATE for S /Smart TV is v.412.419.473
-----------------------------------------------------------
```

Username	Role	Enabled	Access_Code	Password
bob	ROLE_USER	true	$2a$10$Zpb1flWma...	$2a$10$eZeFIPnziokNUi6l...
admin	ROLE_ADMIN	true		$2a$10$nQ798PxvNqsTOE...

Figure 19: Intuitive Flow Diagram of the Heimdallr System (part 2)
This figure gives a closer look at the relationship between the SAA and the Heimdallr along with its admin interface used in Heimdallr system simulation


Heimdallr

Welcome ***admin***

```
Heimdallr ID              : xxxxx
IP Address                : 0.0.0.0/0.0.0.0:3030
Heimdallr WiFi Password   : nxFivwMcS42MO8
```

```
SAA Status                : Connected
Registered Users          : 2
```

Connected Devices

- L[redacted]/Smart Thermostat
- S[redacted]/Smart TV

L[redacted]/Smart Thermostat

```
Device Name        : L[redacted]/Smart Thermostat
Serial No          : 238577384579567834
IP Address         : /127.0.0.1:61940
Firmware Version   : v.465.434.206
Interface Version  : v.331.281.138
Owner(s)           : bob /
```

Device Specific Properties

```
Task Code: 0 -- Content: false
Task Code: 1 -- Content: 0
```

Device Log

```
--Heimdall received:
      Enum type: STATUS
      Device name: L[redacted]/Smart Thermostat
      Task message: [Task code: 0   content: false, Task code: 1    content: 0]
--Heimdall received:
      Enum type: STATUS
      Device name: L[redacted]/Smart Thermostat
      Task message: [Task code: 0   content: false, Task code: 1    content: 0]
--Heimdall received:
      Enum type: STATUS
      Device name: L[redacted]/Smart Thermostat
      Task message: [Task code: 0   content: false, Task code: 1    content: 0]
```

```
--New task received from 238577384579567834:
        • [Task code: 0content: true, Task code: 1    content: 25]
```

S[redacted]/Smart TV

```
Device Name        : S[redacted]/Smart TV
Serial No          : 238577384579034857
IP Address         : /127.0.0.1:61945
Firmware Version   : v.35.266.193
Interface Version  : v.412.419.473
Owner(s)           : bob /
```

Device Specific Properties

```
Task Code: 0 -- Content: false
Task Code: 1 -- Content: 0
Task Code: 2 -- Content: 0
```

Device Log

```
      Enum type: STATUS
      Device name: S[redacted]/Smart TV
      Task message: [Task code: 0   content: false, Task code: 1    content: 0, Task
--Heimdall received:
      Enum type: STATUS
      Device name: S[redacted]/Smart TV
      Task message: [Task code: 0   content: false, Task code: 1    content: 0, Task
--Heimdall received:
      Enum type: STATUS
      Device name: S[redacted]/Smart TV
      Task message: [Task code: 0   content: false, Task code: 1    content: 0, Task
```

```
--New task received from 238577384579034857:
        • [Task code: 0content: true, Task code: 1    content: 10, Task code: 2con
--New task received from 238577384579034857:
        • [Task code: 0content: false, Task code: 1    content: 10, Task code: 2con
```

Figure 20: Intuitive Flow Diagram of the Heimdallr System (part 3)
This figure gives a closer look at the Heimdallr admin interface and its connected device pages used in Heimdallr system simulation

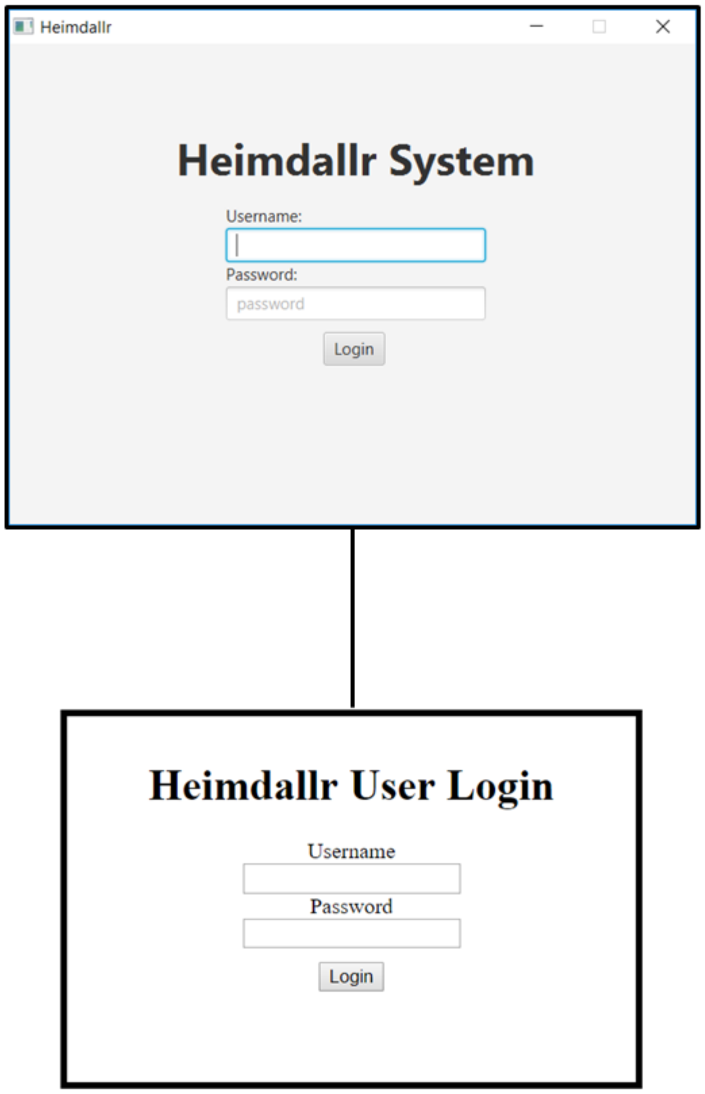
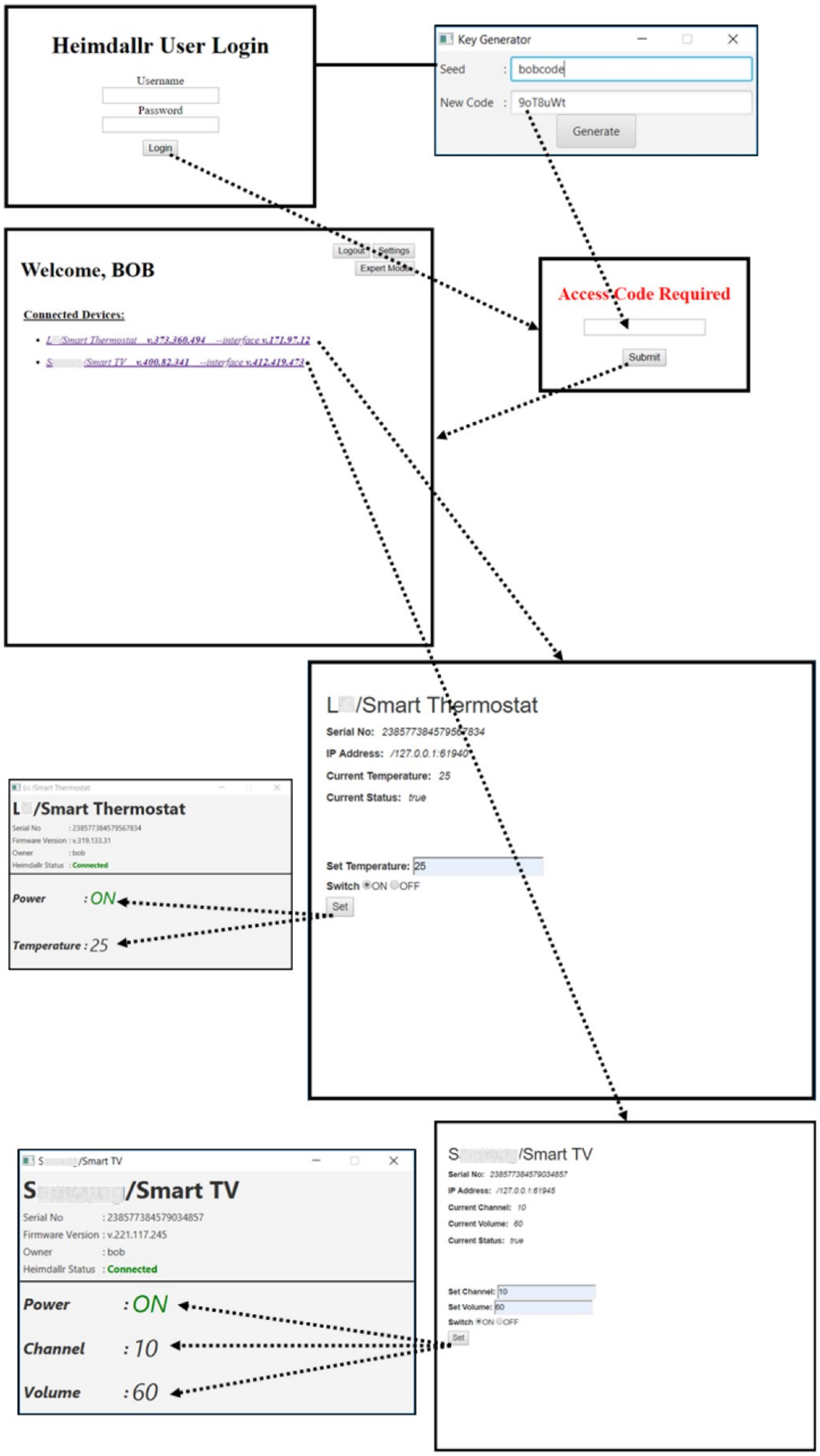


Figure 21: Intuitive Flow Diagram of the Heimdallr System (part 4)
This figure gives a closer look at the relationship between the Heimdallr admin interface and its user web interface used in Heimdallr system simulation



Heimdallr User Login

Username

Password

Key Generator

Seed : bobcode

New Code : 9oT8uWt

Welcome, BOB

Connected Devices:

- L[redacted]/Smart Thermostat v.373.360.494 --interface v.171.97.12
- S[redacted]/Smart TV v.400.82.341 --interface v.412.419.473

Access Code Required

L[redacted]/Smart Thermostat

Serial No: 238577384579567834

IP Address: /127.0.0.1:61940

Current Temperature: 25

Current Status: true

Set Temperature: 25

Switch [✓ ON] OFF

L[redacted]/Smart Thermostat

Serial No : 238577384579567834

Firmware Version : v.319.133.31

Owner : bob

Heimdallr Status : Connected

Power : ON

Temperature : 25

S[redacted]/Smart TV

Serial No : 238577384579034857

Firmware Version : v.221.117.245

Owner : bob

Heimdallr Status : Connected

Power : ON

Channel : 10

Volume : 60

S[redacted]/Smart TV

Serial No: 238577384579034857

IP Address: /127.0.0.1:61945

Current Channel: 10

Current Volume: 60

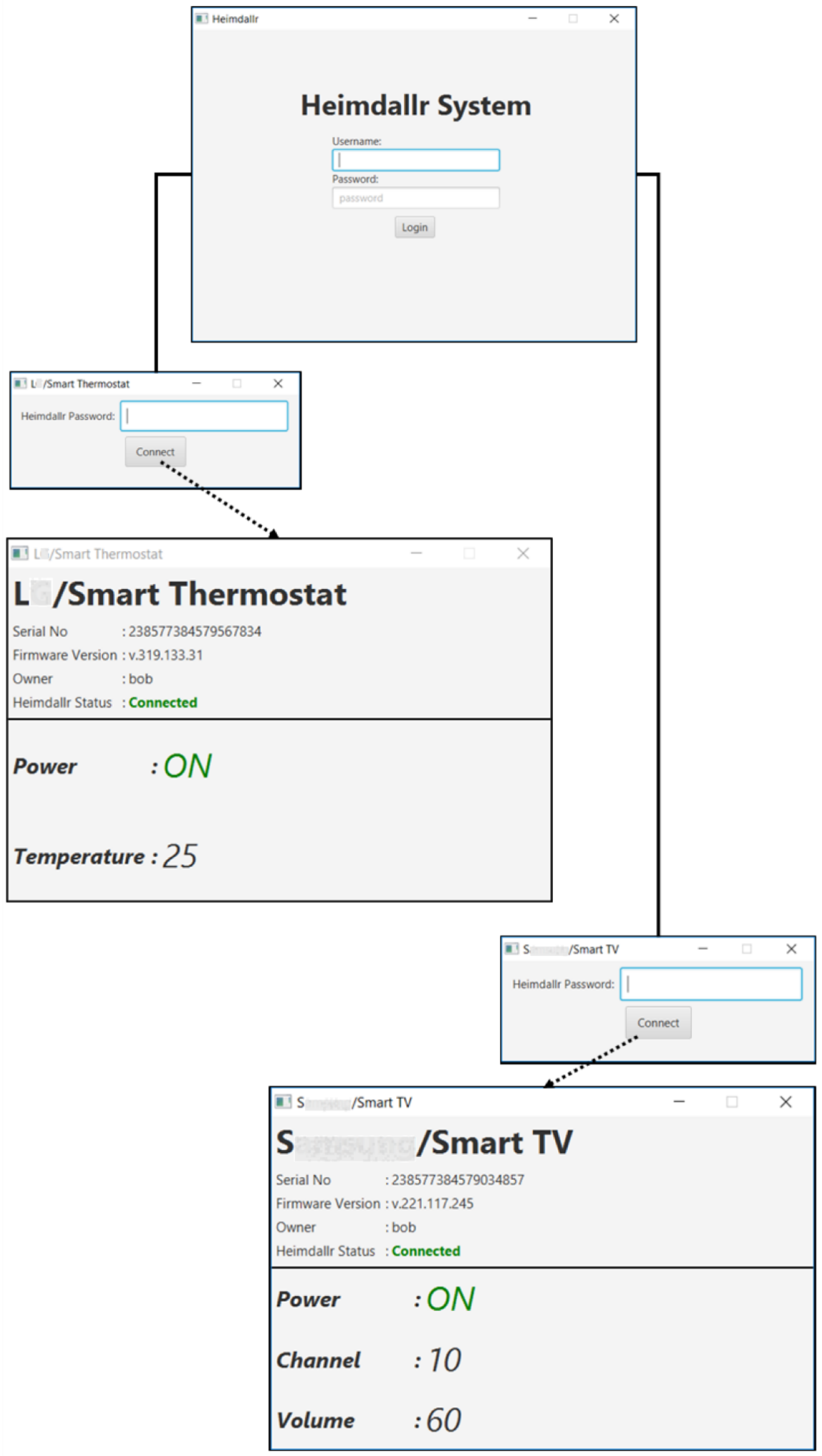
Current Status: true

Set Channel: 10

Set Volume: 60

Switch [✓ ON] OFF

Figure 22: Intuitive Flow Diagram of the Heimdallr System (part 5)
This figure gives a closer look at the relationship between the Heimdallr user web interface and the two IoT devices used in Heimdallr system simulation. It also shows the steps taken by the users to interact with their IoT devices using Heimdallr web interface



L /Smart Thermostat

Serial No : 238577384579567834
Firmware Version : v.319.133.31
Owner : bob
Heimdallr Status : Connected

Power : ON

Temperature : 25

S /Smart TV

Serial No : 238577384579034857
Firmware Version : v.221.117.245
Owner : bob
Heimdallr Status : Connected

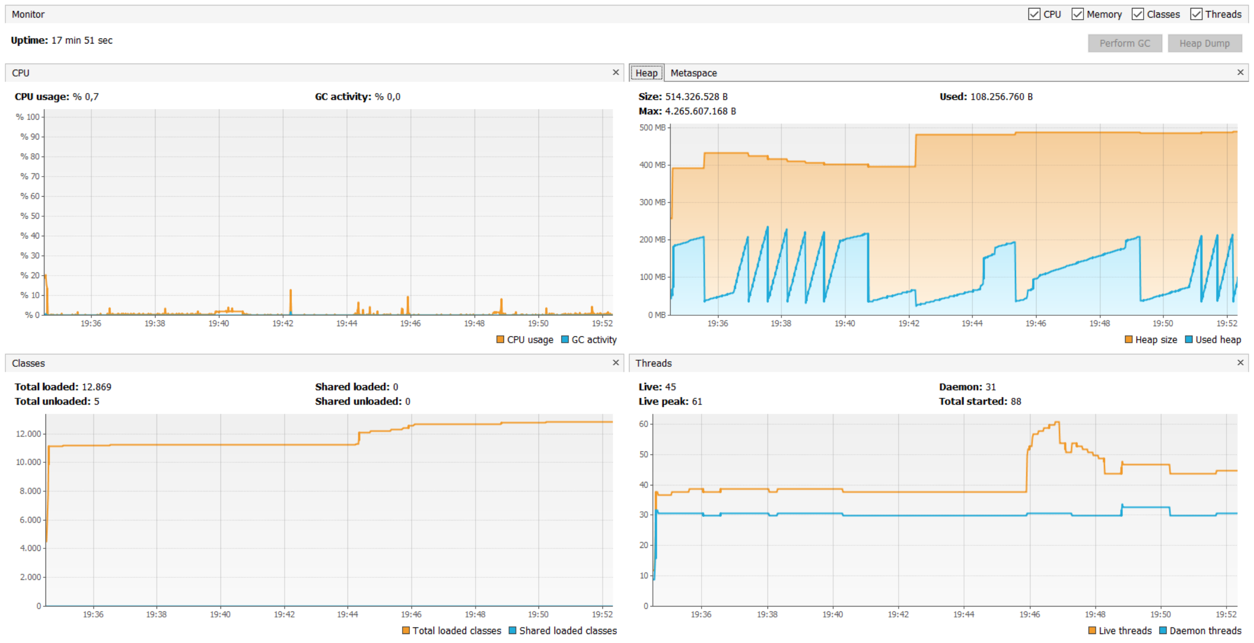
Power : ON

Channel : 10

Volume : 60

Figure 23: Intuitive Flow Diagram of the Heimdallr System (part 6)
This figure gives a closer look at the relationship between the Heimdallr and the two connected IoT devices used in Heimdallr system simulation

8.3 Appendix C



Uptime: 17 min 51 sec

CPU usage: % 0,7
GC activity: % 0,0

Size: 514.326.528 B
Max: 4.265.607.168 B
Used: 108.256.760 B

Total loaded: 12.869
Total unloaded: 5
Shared loaded: 0
Shared unloaded: 0

Live: 45
Live peak: 61
Daemon: 31
Total started: 88

Figure 24: VisualVM Monitor Screenshot of Heimdallr Software
This figure shows CPU, Memory, Class, and Thread data of Heimdallr software gathered using an application analysis tool called VisualVM