IDEMIX BASED ANONYMIZATION FOR HOME AUTOMATION SYSTEMS

by
SIMGE DEMIR

Submitted to the Graduate School of Natural Sciences
in partial fulfilment of
the requirements for the degree of Master of Science

Sabancı University
July 2021

IDEMIX BASED ANONYMIZATION FOR HOME AUTOMATION SYSTEMS

Approved by:

Date of Approval: July 13, 2021

Simge Demir 2021 ©

All Rights Reserved

ABSTRACT

IDEMIX BASED ANONYMIZATION FOR HOME AUTOMATION SYSTEMS

SIMGE DEMIR

Computer Science and Engineering, Master's Thesis, July 2021

Thesis Supervisor: Prof. Albert Levi

Keywords: Internet of Things, Home Automation Systems, Mutual Authentication, Privacy-awareness, Anonymous Identification

Lately, the Internet of Things has been a popular research area within academia as well as in the industry. IoT technology has been widely adopted to industry with a variety of applications. Home automation systems (HAS) which helps homeowners to manage their devices remotely is one of these applications. However, smart homes are vulnerable to various network based attacks. Another important threat in HAS is that it is possible to leak private information of homeowners. In this thesis, we propose a privacy-aware anonymous identification and authentication model for HAS. In the proposed scheme, an innovative gateway is presented to build a secure intercommunication platform between IoT devices and the outside users. Besides, the proposed system is privacy-aware with the introduction of fake proofs which aims to protect users' private information. Anonymity is provided by the Idemix based anonymous credential system where the real identities of the users are hidden. We give implementation details and the results of conducted experiments with downstream and upstream traffic scenarios. Our results suggest that the proposed model is efficient and scalable for home automation systems.

ÖZET

EV OTOMASYON SISTEMLERI IÇIN IDEMIX TABANLI ANONIMLEŞTIRME

SIMGE DEMIR

Bilgisayar Bilimi ve Mühendisliği YÜKSEK LİSANS TEZİ, Temmuz 2021

Tez Danışmanı: Prof. Albert Levi

Anahtar Kelimeler: Nesnelerin Interneti, Ev Otomasyon Sistemleri, Çift Yönlü Doğrulama, Gizliliğe Duyarlılık, Anonim Tanımlama

Son zamanlarda, Nesnelerin İnterneti (IoT) endüstride olduğu kadar akademide de popüler bir araştırma alanı olmuştur. IoT teknolojisi, çeşitli uygulamalarla endüstride yaygın olarak kullanılmaktadır. Ev sahiplerinin cihazlarını uzaktan yönetmelerine yardımcı olan ev otomasyon sistemleri (HAS) bu uygulamalardan biridir. Ancak akıllı evler, ağ tabanlı saldırılara karşı güvenlik açısından zayıftır. Ev otomasyon sistemlerindeki bir diğer önemli tehdit ise ev sahiplerinin kişisel bilgilerinin sızdırılmasının mümkün olmasıdır. Bu tezde, ev otomasyon sistemleri için gizliliğe duyarlı bir anonim tanımlama ve kimlik doğrulama modeli önerilmektedir. Önerilen modelde, IoT cihazları ve dış kullanıcılar arasında güvenli bir iletişim platformu oluşturmak amacıyla yenilikçi bir ev ağ geçidi (IHG - Innovative Home Gateway) sunulmaktadır. Ayrıca, önerilen sistem, kullanıcıların kişisel bilgilerini korumayı amaçlayan sahte kanıtlar ile gizliliğe duyarlı hale getirilmiştir. Kullanıcıların gerçek kimliğinin gizlendiği Idemix tabanlı anonim ehliyet sistemi önerilen modele anonimlik sağlamaktadır. Modelin uygulamasının ayrıntıları ve yukarı yönlü ve aşağı yönlü iletişim senaryoları için yürütülen testlerin sonuçları sunulmaktadır. Sonuçlar, önerilen modelin ev otomasyon sistemleri için verimli ve ölçeklenebilir olduğunu göstermektedir.

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere gratitude to my advisor Prof. Albert Levi for his support and guidance during my research and also my academic life.

Also, I appreciate the participation and valuable feedback of thesis jury members Assoc. Prof. Cemal Yılmaz and Asst. Prof. Kübra Kalkan Çakmakçı.

I will always remember the time we spent on our lab, FENS 2014, with my project partner Şevval Şimşek especially in this pandemic year. I would like to thank to her for being such a supportive friend during my educational life.

This work has been partially supported by TÜBİTAK (Scientific and Technological Research Council of Turkey) under grant 117E017 and I would like to thank to TÜBİTAK members for their support to our project.

Last but not least, I would like to express my gratitude to my family for their endless love and support throughout my life.

TABLE OF CONTENTS

LIST OF TABLES ..... **ix**

LIST OF FIGURES ..... **x**

1. INTRODUCTION **1**

2. BACKGROUND INFORMATION **3**

2.1. Literature review 3

2.2. CL Signature Scheme 4

2.3. Idemix 6

2.4. MQTT 8

3. PROPOSED MODEL **10**

3.1. System Overview 10

3.2. Attacker Model 13

3.3. System Parts 13

3.3.1. Credentials 13

3.3.2. Proof Specification Document 14

3.3.3. Proof 17

3.3.4. Fake Proofs 19

3.4. Privacy-aware Anonymous Identification Model 21

3.4.1. Credential Issuance Protocol 21

3.4.2. Mutual Authentication Protocol 23

3.4.2.1. Protocol Flow of Downstream Traffic Scenario 24

3.4.2.2. Protocol Flow of Upstream Traffic Scenario 27

4. EXPERIMENTAL RESULTS **29**

4.1. Setup and Implementation Details 29

4.2. Performance Evaluation 30

4.2.1. Update Scenario 30

4.2.1.1. Successful update 31

4.2.1.2. Failed update 33
4.2.1.3. Mixed update 36
4.2.2. Error Report Scenario 37
4.3. Discussion 39

5. SECURITY ANALYSIS 41
5.1. Security Analysis 41
5.2. Discussion 45

6. CONCLUSION AND FUTURE WORK 46

BIBLIOGRAPHY 47

LIST OF TABLES

Table 2.1. The symbol definitions used in CL-signature scheme 5

Table 3.1. Symbols used in protocol descriptions..... 11

Table 4.1. Average computational delay results for successful update scenario 31

Table 4.2. Average end to end latency results for successful update scenario 32

Table 4.3. Average end to end latency based on different file sizes for successful update 33

Table 4.4. Average computational delay results for failed update scenario . 34

Table 4.5. Average end to end latency results for failed update scenario... 34

Table 4.6. Average end to end latency based on different file sizes for failed update 35

Table 4.7. Average end to end latency results for mixed update scenario .. 36

Table 4.8. Average computational delay results for error report scenario .. 38

Table 4.9. Average end to end latency results for error report scenario 38

LIST OF FIGURES

Figure 2.1. Overview of Idemix 7

Figure 3.1. Architectural overview of the proposed system 12
Figure 3.2. An example of a credential 15
Figure 3.3. An example of a proof specification document 16
Figure 3.4. An example of a proof 17
Figure 3.5. An example of a fake proof 20
Figure 3.6. Credential issuance protocol 22
Figure 3.7. Mutual Authentication Protocol for Downstream Traffic Scenario 24
Figure 3.8. Mutual Authentication Protocol for Upstream Traffic Scenario 26

Figure 4.1. Average end to end latency for successfully receiving updates 32
Figure 4.2. Average end to end latency for receiving failed updates 35
Figure 4.3. Average end to end latency for mixed updates simultaneously 37
Figure 4.4. Average end to end latency for sending error report 39

Figure 5.1. Mutual authentication protocol simulation using SPAN tool .. 42
Figure 5.2. Mutual authentication protocol intruder simulation using SPAN tool 43
Figure 5.3. Mutual authentication protocol safety analysis with ATSE mode 44
Figure 5.4. Mutual authentication protocol safety analysis with OFCM mode 44

1. INTRODUCTION

The Internet of Things (IoT) is considered as a global network of machines where devices can interact with each other. Recently, IoT research has grown rapidly due to the developing communication technology and easily accessible devices [1]. The adoption of the IoT technology gains high momentum since there is a pressure on the firms to keep up with the technological improvements [2]. There are various kinds of IoT applications in industry such as self-driving vehicles, RFID, indurstrial IoT (IIoT) and home automation systems. Out of these applications, Home Automation Systems (HAS) is one of the most popular areas due to its numerous benefits. In the literature, there are studies [3, 4] that aim to create a secure and efficient home environment using IoT technology.

The aim of HAS is to comfort the homeowners and make their lives easier. HAS includes a set of interconnected devices that can be accessed remotely. For example, in a home automation system, homeowners may control their air conditioner from remote locations or any device at home may send an alert to the homeowner if something goes wrong. However, the communication over the Internet in HAS leads to potential weaknesses. HAS appears as an attractive target for attackers for several reasons [5]. The transferred data in HAS is personal information about homeowners, which should be protected from third parties. Also, the devices are connected to the Internet that makes it easy to attack. Moreover, the devices in home automation systems belong to different vendors and each of them has its own vulnerabilities. As a result, security threats for home automation systems are recently studied and secure and privacy-preserving solutions are being presented [6, 7].

In this thesis, we propose a privacy-aware anonymous identification and authentication protocol for home automation systems as part of the Turkish-Polish bilateral FUSE (Full-Managed Secure Gateway for Home Automation Systems) project. The proposed system is adapted from the HAS architecture of Batalla and Gonciarz [8] which is an outcome of the same FUSE project including Innovative Home Gateway (IHG) and Home Management System (HMS).

In the scheme, IHG is a gateway similar to a set-top box and it handles the communication between outside users and IoT devices. On the other hand, HMS manages the communication by acting as an MQTT [9] broker.

In the proposed model, we ensure anonymity of the users by implementing the Idemix [10] credential system and authenticity by presenting a mutual authentication protocol. We further provide privacy-awareness for the devices by introducing a fake proof system. The contributions can be summarized as follows:

- ***Anonymous Identification:*** Idemix based anonymous credential system is used where the users are known by their pseudonyms and the real identity of the user can be kept anonymous.
- ***Authentication:*** Mutual authentication protocol is introduced where both parties in the communication authenticate themselves by generating proofs based on their credentials.
- ***Privacy-awareness:*** In order to preserve privacy, we present fake proofs which make the entities proceed with the communication in any case and prevent attackers from realizing the failed communication.

Outline of this thesis is organized as follows: In Chapter 2, we give a literature review of IoT research together with background information about Idemix [10], Camenisch-Lysyanskaya (CL) signature scheme [11] and MQTT [9] messaging protocol. In Chapter 3, we explain the proposed anonymous identification model for home automation systems. We give details of the system architecture, parts that are mostly adapted from Idemix and the proposed credential issuance protocol and mutual authentication protocol. Chapter 4 presents the experimental results together with the implementation details. In Chapter 5, we give a security analysis on the mutual authentication protocol. Lastly, in Chapter 6, we conclude the results and propose future work on the subject.

2. BACKGROUND INFORMATION

In this section, first we discuss related work in literature. Then, we present background information about underlying technologies used in the proposed model. To begin with, we provide background information about Idemix [10] which is an anonymous credential system adapted to the proposed scheme for the identification of entities. Together with Idemix, we explain CL-signature scheme [11] that is used in Idemix credential system. After that, we present MQTT [9] which is another technology utilized in our model. Implementation details of Idemix [10] and MQTT [9] protocols will be discussed in Chapter 3.

2.1 Literature review

There exist many studies on secure and anonymous authentication for the Internet of Things. Liu et al. [12] proposed a secure authentication and access control for IoT environment. They present a security analysis on the proposed approach and it is shown that the architecture provides devices from attacks such as eavesdropping, man-in-the-middle attacks and replay attacks. This was an early study which was mainly an analysis of security on the proposed model. Alcaide et al. [13] work on anonymous authentication in a privacy-preserving manner where the only devices that are communicated are the anonymous approved devices. However, there is a weakness in the system at the individual level such that without knowing the private key, an attacker can have access to the system. Also, Lin et al. [14] has proven that this system is insecure since the method of authentication lets an adversary impersonate a legitimate user and deceive the users. In a study by Alizai, Tareen and Jadoon [15], a multi factor authentication scheme is presented. To authenticate a device, they used digital signatures and device capability in the proposed model. If the multi factor authentication does not fail, the device is allowed into the network.

The work by Alizai et al. [15] provides an efficient and less overhead authentication scheme for IoT. A research by Zhang et al. [16] focuses on a secure smart health system which provides aggregate authentication and access control in IoT. In this work, an anonymous certificateless signature scheme is introduced for authentication. In addition to the approach, security analysis and experimental results are presented and the results indicate that the proposed system is efficient in terms of computation and communication cost. In another study by Zhang, Ye and Mu [17], the authors propose a privacy-aware anonymous authentication between user and cloud in smart health systems. To achieve lightweight computation, online-offline techniques are used. As a future work, they suggest that the protocol can be extended to meet higher security and efficiency requirements. On the other hand, Zhou et al. [18] propose an efficient authentication protocol for cloud computing architecture including IoT. To provide efficiency, they use lightweight cryptographic modules such as one way functions. This way, the scheme becomes applicable for the objects that have limited computing power such as IoT devices. By evaluating the performance results, the proposed system is shown as practical and highly suitable for low power devices.

While many privacy-preserving, secure and anonymous identification systems exist for Internet of Things, none of them match with all the security and privacy requirements along with the anonymity of the devices in home automation systems. The contribution of this thesis to the literature is to provide anonymity and privacy-awareness together with secure authentication for IoT devices in a home automation system architecture.

2.2 CL Signature Scheme

The CL signature scheme [11] is developed by Jan Camenish and Anna Lysyanskaya as a signature scheme that is ideal for anonymous credential systems. Idemix [10] credential system developed by IBM Zurich implements the CL signature scheme to accomplish anonymity.

CL signature scheme depends on zero-knowledge proofs [19]. The core idea in CL signatures is proving the knowledge of the master secret without revealing the secret itself.

The issued credential is tied to the master secret but it can be used with unique pseudonyms several times. This property makes the different uses of the credential issued by using CL signature to be untraceable.

Basically, a commitment on the master secret is passed to the Issuer and the issuer signs the committed value by performing zero-knowledge proof protocol. Master secret is blinded in the scheme so that it is kept private. So, CL signature scheme is the building block of the Idemix library which allows the user to prove its attributes without giving no more information than needed. This way, anonymity and security is provided in the Idemix credential system. Definitions of the symbols used in CL signature scheme protocols are given in Table 2.1.

Table 2.1 The symbol definitions used in CL-signature scheme

Symbol	Definition
n	RSA modulus
l_n	size of RSA modulus
$p, q, p^{'}, q^{'}$	prime numbers
R_i, S, Z	quadratic residues modulo n
$\{m_i\}$	message set
l_m	size of the message
e, v	random primes
l_e, l_v	size of random primes
l_r	security parameter

Related protocols of the CL signature scheme are as follows [20]:

- Key Generation: Generate primes $p^{'}$, $q^{'}$ and compute $p \leftarrow 2p^{'}+1$, $q \leftarrow 2q^{'}+1$ and $n \leftarrow pq$ where n is the l_n bit RSA modulus. Then, uniformly randomly choose $R_0, ..., R_{L-1}, S, Z \in QR_n$. Output the public key $(n, R_0, ..., R_{L-1}, S, Z)$ and p as secret key.

- Message space: Let l_m be a parameter. Then the message space is the set $(m_0, ..., m_{L-1})$ where $m_i \in \pm\{0,1\}^{l_m}$.

- Signing algorithm: Let input be $m_0, ..., m_{L-1}$. Choose a random prime e of length $l_e > l_m + 2$, and another random integer v of length $l_v \leftarrow l_m + l_n + l_r$, where l_r is a security parameter. Compute value A as:

$$A \leftarrow \left(\frac{Z}{R_0^{m_0} ... R_{L-1}^{m_{L-1}} S^v} \right)^{1/e} mod\, n \tag{2.1}$$

  The final signature consists of (A, e, v).

- Verification algorithm: In order to verify signature (A, e, v), the followings should hold:

$$Z \equiv A^e R_0^{m_0} ... R_{L-1}^{m_{L-1}} S^v \, (mod\, n) \tag{2.2}$$

$$m_i \in \pm\{0,1\}^{l_m} \tag{2.3}$$

$$2^{l_e} > e > 2^{l_e - 1} \tag{2.4}$$

2.3 Idemix

In security applications, credential systems are extensively used for identification purposes. A user gets the credential in order to prove that it has the requested attributes. However, in such cases where the same credential is used for different scenarios, the usages can be linked to each other [10]. This link may leak information about the user. So, to provide unlinkability, anonymous credentials systems are presented [21, 22]. Basically, in anonymous credential systems, users are known by pseudonyms and for different usages a user may have different pseudonyms which can not be linked. This way, organizations or verifiers know only the fact that the user owns the requested credential.

Idemix [10] is one of the anonymous credential systems developed by IBM. In Idemix architecture there are three parties: Issuer, User and Verifier. Figure 2.1 shows the overview of Idemix credential system.

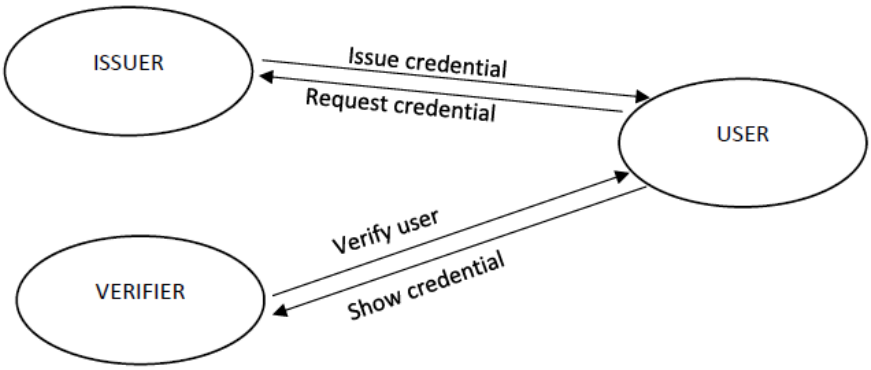

Figure 2.1 Overview of Idemix

Issuer has the authority to issue credentials to the User based on its attributes. Credential issuance is one of the main protocols implemented in the Idemix system. While issuing the credential, CL signature scheme [11] is used which can be considered as the building block of the Idemix system.

In order to run the issuance protocol between the user and the issuer, they need to share the same system parameters. At the end of the issuance protocol, the user receives the credential signed by the issuer which proves that the user has the specified set of attributes. Issuer signs the credential with private key so that any verifier can use issuer public key to verify the credential. Also, the pseudonym of the user is attached to the credential again for the verification process.

At any point, the user may need to prove that she has the specified attributes. Verifier is the entity to check whether the user is correct with her claims. Verification process proceeds with zero-knowledge proof protocol [19]. Using zero-knowledge proof, the user can show that she has the credential containing the set of attributes and that she knows the master secret bound to the pseudonym without revealing the credential which should be kept private. This way, different uses of the same credential can not be linked.

In our proposed scheme, the initialization protocol of idemix is adapted and all the parties in the system get their credentials. Also several functionalities of the system such as proofs, proof specification documents, challenges are adapted from Idemix. Details will be explained in Chapter 3.

2.4 MQTT

MQTT [9] is a widely used messaging protocol for the Internet of Things. It is designed especially for devices which have low processing power such as IoT devices. It is implemented as a lightweight publish/subscribe mechanism minimizing the network bandwidth. The protocol includes a broker where all parties can connect and subscribe to desired topics. When a message is published over a topic, only the subscribed users can receive the message.

In our proposed model, MQTT [9] is preferred for the communication protocol between IHG and Vendor. Also, we choose the broker as Eclipse Mosquitto [23] which is an open-source message broker that implements MQTT protocol.

The working principle of MQTT is to publish and subscribe to the topics. More precisely, any client connected to a MQTT broker can subscribe to a topic and start to listen to messages which are published on that topic. In order to make the communication channel private, topic hierarchy can be established between the parties using a slash (/) separator. For example, update/deviceType is a topic where only the specific type of devices may subscribe and receive the messages published on the topic. This way, other types of devices do not subscribe and as a result can not read the messages. Subscription to a topic may be explicit or may include wildcards (+ or #) [24]. The usages of wildcard characters are as follows:

- "+" is used for a single level of hierarchy. For instance, if the topic is update/+/deviceType then subscribed user receives messages from the following topics:
  - update/version/deviceType
  - update/nonce/deviceType
- "#" is used for all remaining levels of hierarchy. If the subscribed topic is update/deviceType/#, then the user receives all the messages published to the following topics:
  - update/deviceType/version/nonce
  - update/deviceType/nonce

In the proposed scheme, we need a unique topic in order to secure the communication channel so that an adversary can not listen to the exchanged messages. We introduce a topic hierarchy where both parties add randomly generated universally unique identifiers (UUID) [25] to the topic. At first, the parties are subscribed to the topic such as processName/+/+/+. Then, generated UUID pairs are added to the topic on both sides and the final topic hierarchy becomes processName/UUID1/deviceType/UUID2.

3. PROPOSED MODEL

In this section, we give details of the privacy-aware Idemix based anonymization model for home automation systems. First, we present the system architecture to illustrate the communication model. Then, we introduce the system parts that are mostly adapted from Idemix protocols. Lastly, we explain the steps of initialization protocol and the mutual authentication protocol. Symbol definitions that are used in protocol explanations are given in Table 3.1.

3.1 System Overview

In the proposed model, the illustrated architecture for home automation systems originated from Batalla and Gonciarz [8] which was also developed for the FUSE project. It is adapted for the implementation of the proposed anonymous identification and authentication scheme. The adapted scheme is also presented in our paper [28] as part of the FUSE project. Figure 3.1 shows the architecture where there are five entities as Vendor, Homeowners (HO), Home Management System (HMS), Innovative Home Gateway (IHG) and IoT devices.

In this architecture, IoT devices are connected to Innovative Home Gateway (IHG) which is a device similar to a set-top box and provided by network operators to the users. The aim of this gateway is to handle the communication between IoT devices and outside users. IoT devices are first registered to IHG and cannot communicate with any other device. By disabling the direct communication with IoT devices, IHG prevents the devices from network level attacks such as DDos.

Table 3.1 Symbols used in protocol descriptions

Symbol	Description
$IoTD$	IoT device
$UUID_i$	i^{th} generated universally unique identifier [25]
n_i	i^{th} generated nonce
U	Cryptographic attribute structure
S, R_i, Z	A part of issuer's public key
N_{IoTD}	Pseudonym of the user
A	Ordered set of attributes
$(m_i)_{i\epsilon A}$	Attributes in A
g,h	Public common group parameters
m	Master secret
r, $v^{'}$	Random integers
n	RSA modulus for CL-signature [11]
σ_{CL}	CL-signature [11]
$IoTD_A$	List of appropriate devices
$IoTD_C$	List of devices connected to IHG
$(\bar{m}_i)_{i\epsilon A_{\bar{r}}}$	Attributes in the list $A_{\bar{r}}$
$(m_i)_{i\epsilon A_r}$	Attributes in the list A_r
$A_{\bar{r}}$	List of non-revealed attributes of the user
A_r	List of revealed attributes of the user
l_e	Size of e values of certificates
CH_i	i^{th} generated challenge
MSG	Message sent at the end of the mutual authentication protocol
C_{IoTD}	IoT device's certificate
C_V	Vendor's certificate
$Proof_{IoTD}$	generated proof by $IoTD$
$Proof_i$	i^{th} generated proof
$fakeProof_i$	i^{th} generated fake proof
PS	Shared proof specification document
$IDPS$	Id of the PS
$type_{IoTD}$	Device type of $IoTD$
$type_{PS}$	Device type specified in PS
$Hash(.)$	Secure Hash function [26]
$HMAC(.)$	Keyed-Hashing for authentication [27]
$HMAC_{CH_i}$	Generated $HMAC$ using CH_i
$HMAC_{MSG}$	Generated $HMAC$ using MSG
$brand_{IoTD}$	Brand of the IoTD
$comm$	commitment to the master secret

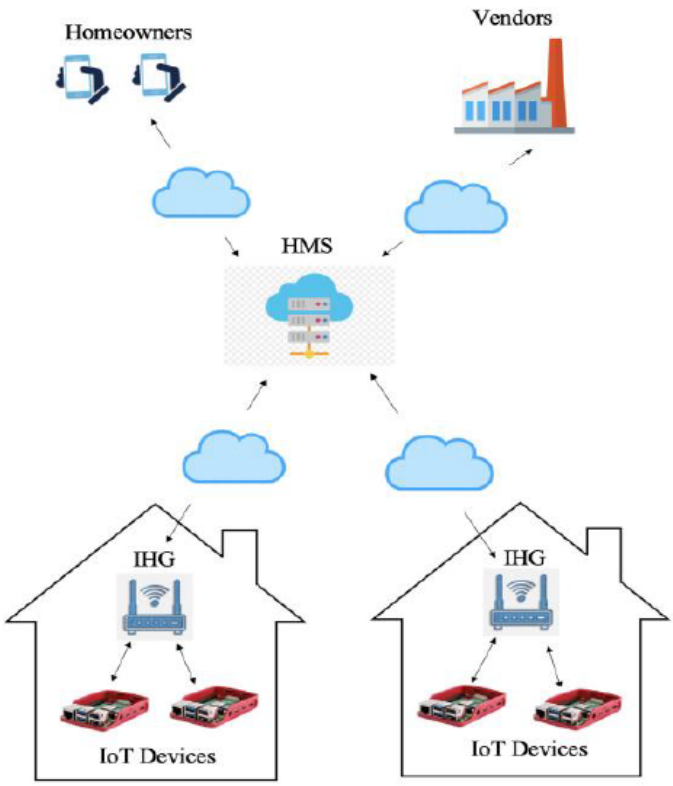

Figure 3.1 Architectural overview of the proposed system [28]

Another entity in the proposed home automation system architecture is Home Management System (HMS) which has two different roles. It is the issuer where the credentials are issued for the entities in the system as well as the MQTT [9] broker that manages the communication between Vendor and IHG. HMS belongs to the network operator which distributes IHG set-top boxes.

In the model, communication can start any time. For a particular reason, Vendors or the Homeowner may want to communicate with IoT devices. Vendors may send periodic updates or homeowners may want to give commands to an IoT device remotely. On the other hand, when an IoT device encounters a problem, it sends an error report to inform the Vendor about the situation. With an intermediary device (IHG), outside users can only send their messages to IHG and IHG will relay the messages to IoT devices to build a secure intercommunication.

3.2 Attacker Model

In this section, we introduce attacker model that clarifies the capabilities of an attacker in the system. To begin with, an attacker can listen to the channel and capture the messages by performing eavesdropping attack. Attacker can also modify bits in the message or replace the message and impersonate any party in the system as in man-in-the-middle attack scenario. However, if the messages are encrypted, the attacker cannot decrypt it and cannot obtain the original message unless it does not have the secret key.

On the other hand, attacker can perform denial of service (DoS) attacks by flooding the target with high amount of requests to make the network become unavailable. However, if the devices are closed to network communication then it cannot perform DoS attack to these devices in the system. Moreover, an attacker can measure the latency values and compare the results in order to have an insight based on different communication scenarios. However, attacker cannot authenticate itself in the system if it is not a legitimate user.

3.3 System Parts

In this section, we present the system parts that are mostly stemmed from Idemix [10]. We have adapted them to use in the proposed model. They are used in XML structure and can be parsed by a parser implemented in Idemix library [20].

3.3.1 Credentials

Credential is the document that states the user has any or all of the subset of the claimed attributes. Any verifier may ask a user to prove its identity in the system. Credentials are used to generate proofs indicating that the user has the desired attributes in the credential signed by an issuer.

In the proposed protocol, anonymity comes from the credential system. We implemented the Idemix anonymous credential system where the users are only known by their pseudonyms and different usages of the credential cannot be linked to each other. This way, we provide anonymous identification of the users in the system.

The XML structure of an example credential is shown in Figure 3.2. As seen in the figure, a credential includes attributes of the IoT device such as brand, type and model. The other important part in the credential is the signature which is issued by a legitimate issuer. Issuer public key location is also attached to the credential so that any verifier can validate the credential signed by the issuer using its private key.

In the proposed home automation system architecture, HMS has the responsibility of an Issuer and IoT devices, Vendors and Homeowners are the Users which first register to the system and request their credentials from HMS. Further details of credential issuance protocol is explained later.

3.3.2 Proof Specification Document

Proof specification document (proofSpec) is the document containing the attribute list that is to be proven. The user needs to prove that all the attributes in the proofSpec also exist in the user's credential. On the other hand, the Verifier needs proofSpec to know which attributes are proven.

In the proposed model, proofSpec document is chosen by the verifier beforehand and the name of the proofSpec is shared with the User so that they agree on a document including desired attributes. All the proofSpec documents are reachable on a public server.

An example proof specification document can be seen in Figure 3.3. As shown in the figure, the credential structure that is used in the verification process is indicated in the proofSpec. Also, it includes the specific attributes that the device should prove to have. In the example of Figure 3.3, in order to be verified, the IoT device's brand needs to be "Arcelik" and it should be a lamp.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Credential xmlns="http://www.zurich.ibm.com/security/idemix"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
http://www.zurich.ibm.com/security/idemix ../xsd/Credential.xsd">
  <References>
    <IssuerPublicKey>http://10.36.3.164:8080/files/issuerData/
    ipk.xml</IssuerPublicKey>
    <CredentialStructure>http://10.36.3.164:8080/files/issuerData
    /CredStruct_IoTDevice.xml</CredentialStructure>
  </References>
  <Attributes>
    <Attribute name="peActorType">
      <Value>73</Value>
      <EnumValue>actorType;IoTDevice</EnumValue>
    </Attribute>
    <Attribute name="peBrand">
      <Value>7</Value>
      <EnumValue>brand;Arcelik</EnumValue>
    </Attribute>
    <Attribute name="peType">
      <Value>29</Value>
      <EnumValue>type;Television</EnumValue>
    </Attribute>
    <Attribute name="model">
      <Value>-246728037529372992801469033815015107540827695554735
      485636429989094801518833 8</Value>
    </Attribute>
    <Attribute name="homeID">
      <Value>5</Value>
    </Attribute>
  </Attributes>
  <Signature>
    <A>292658817898637340484663478280962248410134779662608308218 2
    310061917940898239444025727981249310349159385183807549585526 3
    625371104838233479518353999562355228591051621179627912157289 7
    169741945763754485880607093603934343767039074821445674525189 3
    15125459134589
    064489522178884908954542192352269902412331699304653 34</A>
    <e>259344723055062059907025491480697571938277889515152306249 7
    285831056658007133067591499816905591939871430970635766985030 2
    763390704083839928199170118780738951694590431117777079936137 7
    </e>
    <v>341530409923534728961351938750514455936080109014759358554 1
    854798855895892749598044170969148524119252839896390491667301 0
    302205148551398804670105009115412986978673699784786395822264 9
    730501169328343027676032846519827645920537771460254975261149 8
    809961609633332501743607190550630325388215667559562914015534 3
    566133424529677910357554559087020365317151539968306637154460 7
    422447528143329691932967895633143485806327744059531371772853 0
    965728</v>
  </Signature>
  <Features/>
</Credential>
```


Figure 3.2 An example of a credential

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ProofSpecification xmlns="http://www.zurich.ibm.com/security/idemix"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
                    "
                    xsi:schemaLocation="http://www.zurich.ibm.com/
                    security/idemix ../xsd/ProofSpecification.xsd">

    <Declaration>
        <AttributeId name="id1" proofMode="unrevealed" type="string" />
        <AttributeId name="id2" proofMode="unrevealed" type="int" />
        <AttributeId name="id3" proofMode="unrevealed" type="enum" />
        <AttributeId name="id4" proofMode="unrevealed" type="enum" />
        <AttributeId name="id5" proofMode="unrevealed" type="enum" />
    </Declaration>

    <Specification>
        <Credentials>
            <Credential issuerPublicKey="http://10.36.3.164:8080/files/
            issuerData/ipk.xml"
                        credStruct="http://10.36.3.164:8080/files/
                        issuerData/CredStruct_IoTDevice.xml" name="
                        IoTDevCred">
                <Attribute name="model">id1</Attribute>
                <Attribute name="homeID">id2</Attribute>
                <Attribute name="peBrand">id3</Attribute>
                <Attribute name="peType">id4</Attribute>
                <Attribute name="peActorType">id5</Attribute>
            </Credential>
        </Credentials>

        <EnumAttributes>
            <EnumAttribute attributeId="id3" operator="and">
                <EnumValue attributeName="brand">Arcelik</EnumValue>
            </EnumAttribute>
            <EnumAttribute attributeId="id4" operator="and">
                <EnumValue attributeName="type">Lamp</EnumValue>
            </EnumAttribute>
            <EnumAttribute attributeId="id5" operator="and">
                <EnumValue attributeName="actorType">IoTDevice</
                EnumValue>
            </EnumAttribute>
        </EnumAttributes>

        <Inequalities />

        <Commitments />

        <Representations />

        <Pseudonyms />

        <VerifiableEncryptions />

        <Messages />

    </Specification>

</ProofSpecification>
```

Figure 3.3 An example of a proof specification document

3.3.3 Proof

Proof is a non-interactive statement [10] which the user generates in a verification process. In the proposed protocol, a legitimate User creates a proof and sends it to Verifier in order to be authenticated. The structure of an example Proof is given in Figure 3.4.

As seen in Figure 3.4, generated proof includes challenge received from the verifier and also common values, s-values and t-values that are generated during proof generation algorithm.

In our model, proof generation (buildProof) and verification (verifyProof) protocols are adapted from Idemix library [20]. Algorithm 1 shows the steps of proof generation of an IoT device while Algorithm 2 explains the proof verification steps for a $Proof_{IoTD}$.


```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<IdmxProof xmlns="http://www.zurich.ibm.com/security/idemix" xmlns:xs="
http://www.w3.org/2001/XMLSchema" xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
http://www.zurich.ibm.com/security/idemix ../xsd/IdmxProof.xsd">
  <Challenge>344902053824458549938469800985541023087480604392662023529522 4
  5544105651402401</Challenge>
  <CommonValues>
    <CommonValue key="gp">7848528048562573321140279353432344932111789871 00
    8854384674755983183475943 1000</CommonValue>
    <CommonValue key="http://10.36.3.164:8080/files/issuerData/
    CredStruct_IoTDevice.xml;IoTDevCred">2261784019484972346374971511296 66
    5897643637111799369111032729156743130304219582055786889163172802089542
    8911549986700754174172023047977897889847847372065824201562628661325806
    5928358642557195676782680313526961061090508739103536358506214983512176
    4559579517246443615076986122878698336516635403790817671712723272 2</
    CommonValue>
    <CommonValue key="http://10.36.3.164:8080/files/issuerData/
    CredStruct_IoTDevice.xml;IoTDevCred;issuerPk">5196857095230938150669 17
    9090192094073668529049570749024720958180255637963628 1</CommonValue>
    <CommonValue key="id3;AND">6293387841868028079228107702999009860891170
    7311340394857772281604977474908947417316265134869389764739466677157668
    5247317803070234551547399828045686852051569051059460278111246584827769
    9521090955120612255217751625971187335871713911952981385547783934938490
    3959236499722445799393343256059629590967690132574701480</CommonValue>
    <CommonValue key="id4;AND">1237567978621128948830291489839080568613287
    5580292411990689582158223375073978712395187599661797007590416271748 69
    8155193388592268083734018592449281863410015937057681066913787274165094
    6771079161246403727026975104636417178591385999316206886807280381387393
    3391550430948896756789205729154801258960786474426246352 3</CommonValue>
    <CommonValue key="id5;AND">1468372702498207250803082993703421119570690
    7112158900994649792932878085588271048345693426722760230840870746127629
    1283909365240065557272066028816925206353979483503577580405207224346341
    3529572866798774337948224655043303347463833811009654368775158832825601
    1511362975074308396861332045852162060831434028578108501 5</CommonValue>
    <CommonValue key="proofSpecHash">8303005722223167196466291892534824047
    3414511251707883084009328164096686621836</CommonValue>
    <CommonValue key="sp">1931345493401549450285337266048536427708885195 49
    7901126733189802526692987 1326</CommonValue>
  </CommonValues>
```


Figure 3.4 An example of a proof

Algorithm 1: Proof Generation Algorithm

Input: $comm$, C_{IoTD}, S, n_1, N_{IoTD}
Output: CH_1, s and $common$
if C_{IoTD} *not exists* **then**
  Request C_{IoTD} from Issuer;
else
  Load C_{IoTD} inputs ;
  Randomize $A^{'} := AS^r \, mod \, n$ and save as $common$;
  Compute t-value $t := (A^{'})^r (\prod R_i^{\bar{m}_i})(S^r) \, mod \, n$ and store to the list;
  Compute challenge $CH_1 := Hash(common, t, n_1, comm, N_{IoTD})$;
  Compute s-values $s := r + CH_1 \bar{m}_i$;
  Output $Proof_{IoTD}$ as $(CH_1, s, common)$;
end

In Algorithm 1, first it checks if the credential of IoT device, C_{IoTD}, already exists. If that is not the case, then IoT device, $IoTD$, requests C_{IoTD} from the Issuer. Otherwise, it proceeds with proof generation. Inputs to the algorithm are $comm$, C_{IoTD}, S, n_1, N_{IoTD} where $comm$ is the commitment to master secret, m, n_1 is a nonce sent by the verifier, N_{IoTD} is the pseudonym of the device and S is a part of Issuer's public key. Protocol starts with a randomization of $A^{'} := AS^r \, mod \, n$ where A is a part of CL-signature, σ_{CL}, attached to C_{IoTD}. The resulting value $A^{'}$ is saved to a list $common$, to which the prover and verifier have access. After that, t-values, t, are calculated, using $(\bar{m}_i)_{i \epsilon A_{\bar{r}}}$'s from the list of non-revealed attributes $A_{\bar{r}}$, to be used in the challenge. Also, they are stored to a list accessible by prover and verifier. Calculated challenge, CH_1, is the $Hash(.)$ of generated values as $CH_1 := Hash(common, t, n_1, comm, N_{IoTD})$. Responses to attributes, $(\bar{m}_i)_{i \epsilon A}$, are calculated based on created CH_1 with the formula $s := r + CH_1 \bar{m}_i$ where r is a random integer. Finally, the output is proof of $IoTD$, $Proof_{IoTD}$, including CH_1, s and $common$.

In Algorithm 2, when the verifier receives $Proof_{IoTD} = (CH_1, s, common)$, first it retrieves common value $A^{'}$, s and t. Then, $\bar{t}$-value is computed using received CH_1, retrieved t-value and $A^{'}$ and the attributes $(m_i)_{i \epsilon A_r}$ in the revealed attributes list A_r. After that, a challenge, CH_2, is computed by getting the $Hash(.)$ of the values as $CH_2 := Hash(common, \hat{t}, n_1, comm, N_{IoTD})$ where n_1 is an input to the algorithm. Last step of the algorithm is to check if the received CH_1 is equivalent to the computed challenge CH_2. If not, the $Proof_{IoTD}$ is not verified and rejected. Otherwise, $Proof_{IoTD}$ is verified and accepted.

Algorithm 2: Proof Verification Algorithm

Input: $Proof_{IoTD} = (CH_1, s, common), n_1$
Output: Accept or reject $Proof_{IoTD}$
if *received* $Proof_{IoTD}$ *is null* **then**
  Prompt an error message and stop;
else
  Retrieve *common* value $A^{'}$, t-values t and s-values s;
  Compute $\hat{t}$-values as

$$\hat{t} := \left(\frac{Z}{\prod R_i^{m_i}(A')^{l_e-1}}\right)^{-CH_1} (A^{'})^r (\prod R_i^{\bar{m}_i})(S^r) \, mod \, n$$

  Compute challenge as $CH_2 := Hash(common, \hat{t}, n_1, comm, N_{IoTD})$;
  if $CH_1 \equiv CH_2$ **then**
    Accept $Proof_{IoTD}$
  else
    Reject $Proof_{IoTD}$
  end
end

3.3.4 Fake Proofs

In the proposed anonymous identification model, we introduce fake proofs in order to provide privacy-awareness. Fake proofs are dummy documents that look like legitimate proofs in appearance. In the case of sending updates to IoT devices, an IoT device needs to authenticate itself. If the desired attributes list in the specified proofSpec is not a subset of the attributes in the credential of the IoT device, then that device is not a target to receive updates from the Vendor. To exemplify, if the Vendor is sending updates to televisions and if the IoT device is a television but does not belong to the corresponding Vendor, still the request is sent to that device and the communication is started. When the IoT device receives proofSpec, it realizes that it should not receive that update since it does not satisfy the conditions. However, the protocol execution is not stopped there. Even though the device is not the target for update, it creates a fake proof in order to proceed with the protocol. When the Vendor receives fake proof, it responds with another fake proof. Thus, the communication seems as usual and third parties cannot gain any information about IoT devices's attributes such as brand name. The protocol becomes privacy-aware with the introduction of fake proofs.

An example of fake proof structure is given in Figure 3.5. As shown in the figure, the fake proof structure looks like a legitimate proof. However, the challenge is generated at random and the common values, s-values and t-values are taken from

a preshared dummy document. So, the verification fails and the verifier knows that it is a fake proof.


```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<IdmxProof xmlns="http://www.zurich.ibm.com/security/idemix" xmlns:xs="
http://www.w3.org/2001/XMLSchema" xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
http://www.zurich.ibm.com/security/idemix ../xsd/IdmxProof.xsd">


    <Challenge>615587039717066171598825036158817597967581892463439181791437
    503444434085684878</Challenge>

    <CommonValues>


        <CommonValue key="http://10.36.3.164:8080/files/issuerData/
        CredStruct_IoTDevice.xml;IoTDevCred;issuerPk">5196857095230938150 6
        691790901920940736685290495707490247209581802556379636281</
        CommonValue>


        <CommonValue key="gp">7848528048562573321140279353432344932111789 8
        710088543846747559831834759431000</CommonValue>


        <CommonValue key="proofSpecHash">8303005722223167196466291892534 82
        404734145112517078830840093281640966862183 6</CommonValue>


        <CommonValue key="sp">1931345493401549450285337266048536427708885 1
        954979011267331898025266929871326</CommonValue>


        <CommonValue key="http://10.36.3.164:8080/files/issuerData/
        CredStruct_IoTDevice.xml;IoTDevCred">7265320241458957919278889294 3
        569922065194739266589908068071209022207167237010458939326556684177
        816767091668264800711599702247750373927671455934339483795049167808
        627916157545055272847685920028710190895732988624755818056796083512
        385715482045188792586740616 3
        032953762099320730538227738263898286564290033834426 30</CommonValue
        >
        <CommonValue key="id3;AND">3281497722622938110975061950371993804 05
        695718332556056576319062019525399879911391512593276694060259797202
        237038953166522417464580600466089241550617702929408966438029424055
        933776060573508907435053799748852475604266592969950930368084338207
        653033773570473385
        829642167911689423295783523577413153053265329924706 98</CommonValue
        >
        <CommonValue key="id4;AND">1404842189533585679125385278168854903 62
        864488487822037563083161705648456478070005290572857841106768749341
        984069959685962581365011888236018510980559714074669158127583736414
        499100433024329323853749246606235743152199731039531900286003416647
        895068591887202541
        741103406573295228955249178929027319908192931086700991</
        CommonValue>
        <CommonValue key="id5;AND">8591158447481143525461462206908548593 22
        892040012434842321724985231370377013971639834180145922625850659304
        467418336937344023640807484304264464645094929147419305078818439091
        650190684469304124915874273717300818690817156149425837379905547183
        343868017500376571
        982508564228447209326015019898147783571047187086354 37</CommonValue
        >
    </CommonValues>
```


Figure 3.5 An example of a fake proof

3.4 Privacy-aware Anonymous Identification Model

In this section, we give details of privacy-aware anonymous identification model. In the proposed scheme, IHG communicates with IoT devices and it is the only entity that can connect to the devices. Data flow between Vendor and IHG goes over HMS which is the MQTT broker in the architecture. Vendor and IHG are subscribed to related topics so that they start listening to the other parties while they are connected to the network. On the other hand, IHG relays the published messages to IoT devices over TCP connection which creates an intercommunication at home environment.

When IoT devices first connect to IHG, they run credential issuance protocol and get the credentials for future usages. This credential issuance protocol is adapted from Idemix which is the anonymous credential system as explained in Chapter 2.

After getting the issued credentials, both parties (Vendor and IoT devices) become ready for the communication. From this point on, they can run a mutual authentication protocol anytime needed. More specifically, if an IoT device wants to send an error report to the Vendor or if Vendor needs to send periodic updates to the devices, mutual authentication protocol is performed. In the following sections, flow of both credential issuance and mutual authentication protocols are given.

3.4.1 Credential Issuance Protocol

Initially, entities in the proposed model need to obtain their credentials to be able to identify themselves. Credential issuance protocol is adapted from the Idemix [10] credential system which provides anonymity in the proposed model. By adapting Idemix, anonymous credentials are issued where the users are only known by their pseudonyms and different usages of credentials cannot be linked to each other. This way, we provide anonymous identification of the users in the system.

In the proposed scheme, HMS (as Issuer) generates credentials for IoT devices and Vendors. After performing credential issuance protocol, credentials are signed as CL-signatures and saved at the user's (IoT device or Vendor) side. The protocol flow for credential issuance is shown in Figure 3.6.

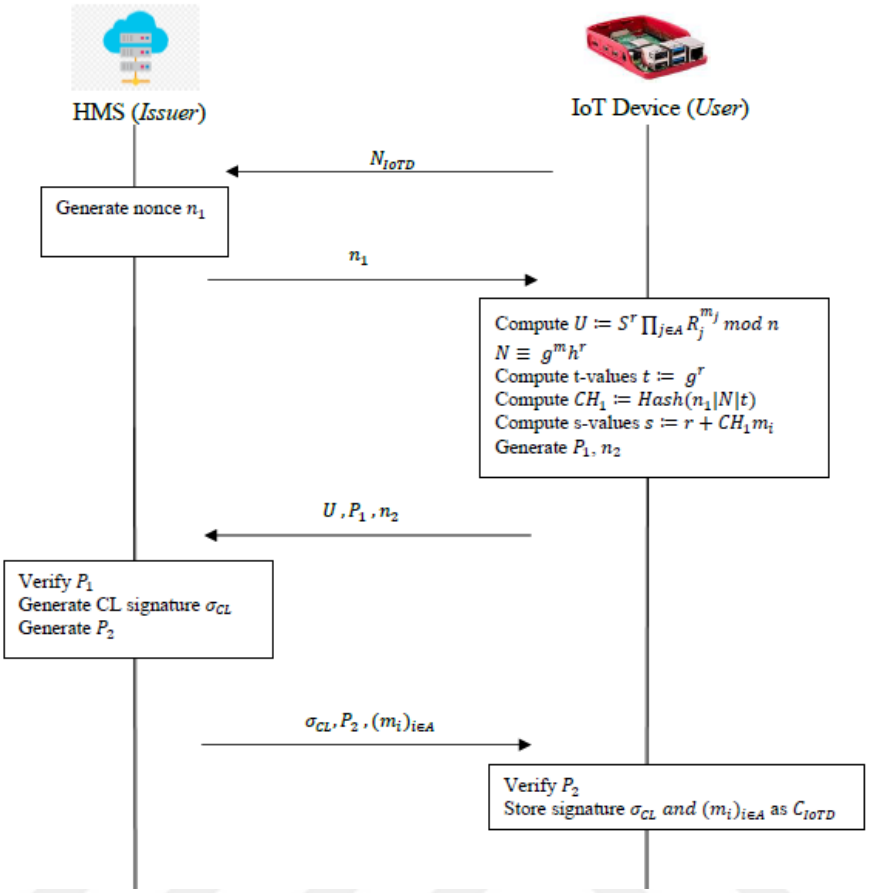

Figure 3.6 Credential issuance protocol

The steps of the credential issuance protocol between Issuer and IoT device are as follows:

- The user ($IoTD$) is known by its pseudonym, N_{IoTD}, in the system. So, the protocol is initialized by sending N_{IoTD} to the issuer.
- When issuer receives a request from $IoTD$ by getting N_{IoTD}, it generates a random nonce, n_1, to proceed with zero-knowledge proof protocol [19]. Issuer sends n_1 to $IoTD$.
- After receiving n_1, $IoTD$ computes a cryptographic attribute structure, U, depending on certain set of attributes, A. Additionally, random t-value, $t = g^r$, are computed and a challenge, CH_1 is generated as $CH_1 = Hash(n_1, N, g^r)$ where $Hash(.)$ is the cryptographic hash function and N is the commitment of m calculated as $N = g_1^m h^r$. Lastly, response as s-value is calculated as $s = r + CH_1 m_1$. $IoTD$ generates a $Proof_1$ to prove the knowledge of master secret m associated with N_{IoTD} by using calculated CH_1 and s. U, $Proof_1$ and nonce n_2 are sent to issuer.

- Issuer first verifies $Proof_1$ to make sure that the user is legitimate and knows the secret m. Then, it also generates $Proof_2$ to identify authenticate to $IoTD$. Additionally, issuer signs the attributes, $(m_i)_{i\epsilon A}$,with a CL-signature σ_{CL} and sends it to $IoTD$.
- $IoTD$ receives the $Proof_2$ and verifies it. If verification is successful, then σ_{CL} is stored at $IoTD$.
- By running the credential issuance protocol, $IoTD$ proves the knowledge of master secret m without revealing it and N_{IoTD} is linked to the credential of $IoTD$, C_{IoTD}.

3.4.2 Mutual Authentication Protocol

In the proposed privacy-aware anonymous identification model, the communication between IoT devices and outside users start with verification processes. Both parties need to authenticate themselves by using their legitimate credentials. This way, mutual authentication is ensured. Privacy-awareness in the mutual authentication protocol is provided by fake proof creation. With the introduction of fake proofs, the communication proceeds in an expected way and as a result, it does not reveal identities of the users.

Mutual authentication protocol can be run between any party whenever it is needed. The protocol is presented for two-way communication model: upstream and downstream traffic. An outside user such as Vendor may send periodic updates to the IoT devices which correspond to downstream traffic scenario or an IoT device may face a problem and want to send an error report to inform the Vendor which is an example of upstream traffic scenario. Figure 3.7 shows the protocol flow for downstream traffic and Figure 3.8 shows the protocol flow for upstream traffic.

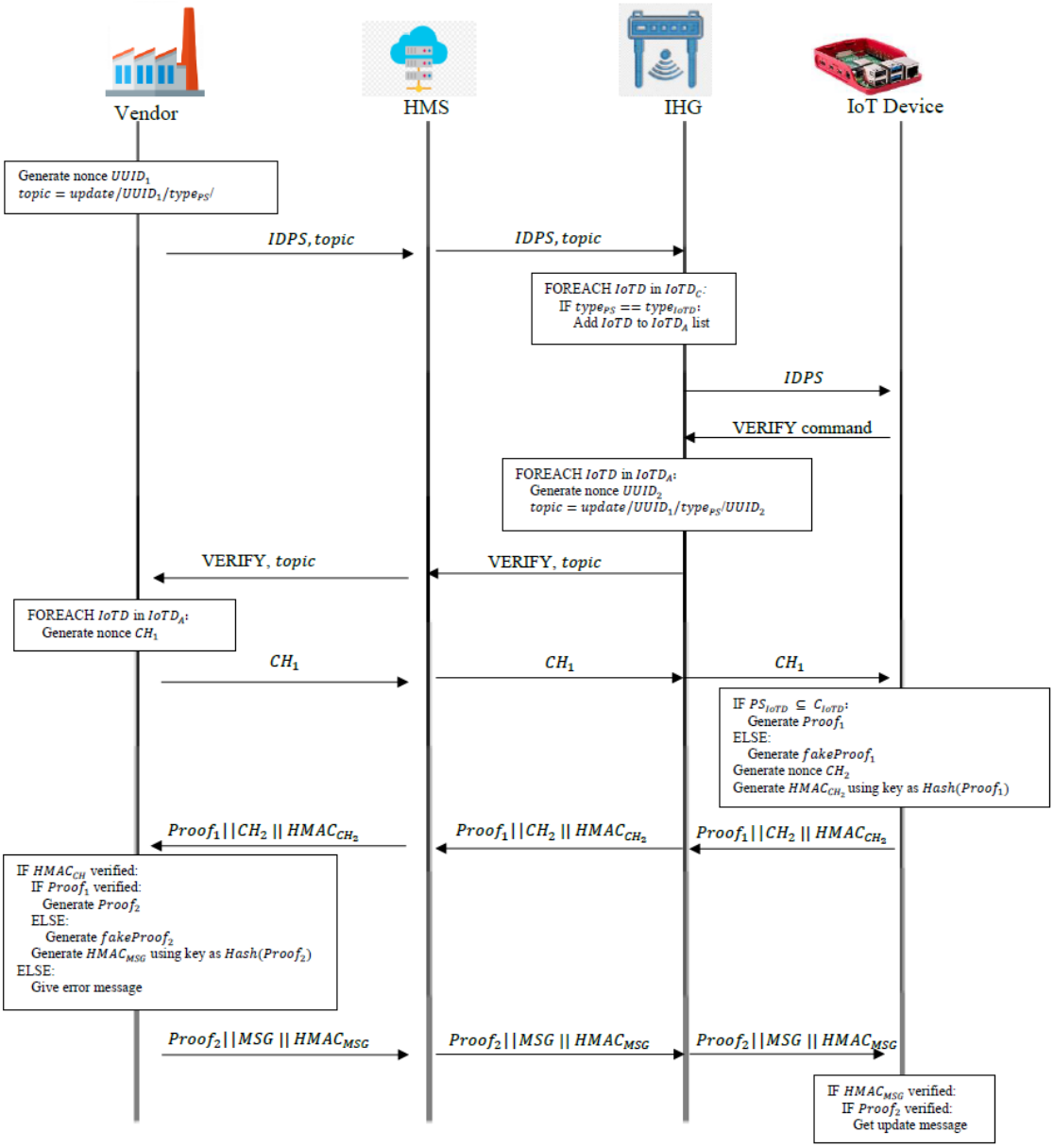

Figure 3.7 Mutual Authentication Protocol for Downstream Traffic Scenario

3.4.2.1 Protocol Flow of Downstream Traffic Scenario

The steps of the mutual authentication protocol in the direction of downstream traffic are as follows:

- At first, IHG is subscribed to the topic $update/+/+$ on which an outside user can publish the first message.

- Developed protocol is initialized by Vendor by choosing the relevant proofSpec, PS, which denotes the attributes of the target device. Vendor also generates a random universally unique identifier, $UUID_1$, to construct the specific topic and sends $IDPS$ to IHG through HMS, the MQTT server.

- After receiving the message of $IDPS$ on the specific topic $update/UUID_1/type_{PS}$, IHG checks the list of all connected devices, $IoTD_C$, and generates the list of appropriate devices, $IoTD_A$, to send the message. $IoTD_A$ consist of the devices where the device type specified in PS, $type_{PS}$, matches with the type of $IoTD$, $type_{IoTD}$.

- When $IoTD$ receives the message $IDPS$, it sends IHG the $VERIFY$ command indicating that in order to proceed, verification is necessary.

- IHG generates $UUID_2$ for each of the $IoTD$ in $IoTD_A$ to create a unique topic for the communication. This way, only IHG and Vendor knows the specific topic to publish the messages. The rest of the messages in the communication are sent over that topic. After this point, only responsibility of IHG is to relay the messages between Vendor and $IoTD$ as it is the gateway in the proposed system. Rest of the messages are published on the newly constructed topic, $update/UUID_1/type_{PS}/UUID_2$.

- When Vendor receives a $VERIFY$ command on the topic $update/UUID_1/type_{PS}/UUID_2$, it generates a random challenge CH_1 and publishes to the topic.

- IHG receives and relays the CH_1 to the $IoTD$. $IoTD$ generates $Proof_1$ as in Algorithm 1 to authenticate itself to Vendor. If the attributes in PS does not match with the attributes in credential of $IoTD$, C_{IoTD}, then $IoTD$ creates a dummy document as $fakeProof_1$ which is seemingly legitimate. Additionally, CH_2 is generated by $IoTD$ to send Vendor in order to provide mutual authentication. To preserve integrity, HMAC of CH_2, $HMAC_{CH_2}$, is sent together with CH_2 and $Proof_1$. In the calculation of $HMAC_{CH_2}$, $Proof_1$ is used as the key string.

- Vendor first verifies $HMAC_{CH_2}$ to check the integrity of the message and if verification does not fail, it also verifies $Proof_1$ to authenticate $IoTD$. If $Proof_1$ can not be verified, Vendor maintains the communication by creating $fakeProof_2$. Otherwise, Vendor generates $Proof_2$ in order to authenticate

itself. Now, the update message, MSG can be sent to $IoTD$. So, Vendor publishes $HMAC_{MSG}$, MSG and $Proof_2$ to IHG.

- IHG sends the published messages to $IoTD$ and first $HMAC_{MSG}$ is verified in order to check the validity of the update message MSG. Then, $Proof_2$ is verified to authenticate and trust the Vendor before downloading the MSG. When both of the verification processes are successful, mutual authentication is ensured and the update is installed on $IoTD$. Otherwise, it is not installed and the protocol execution is stopped.

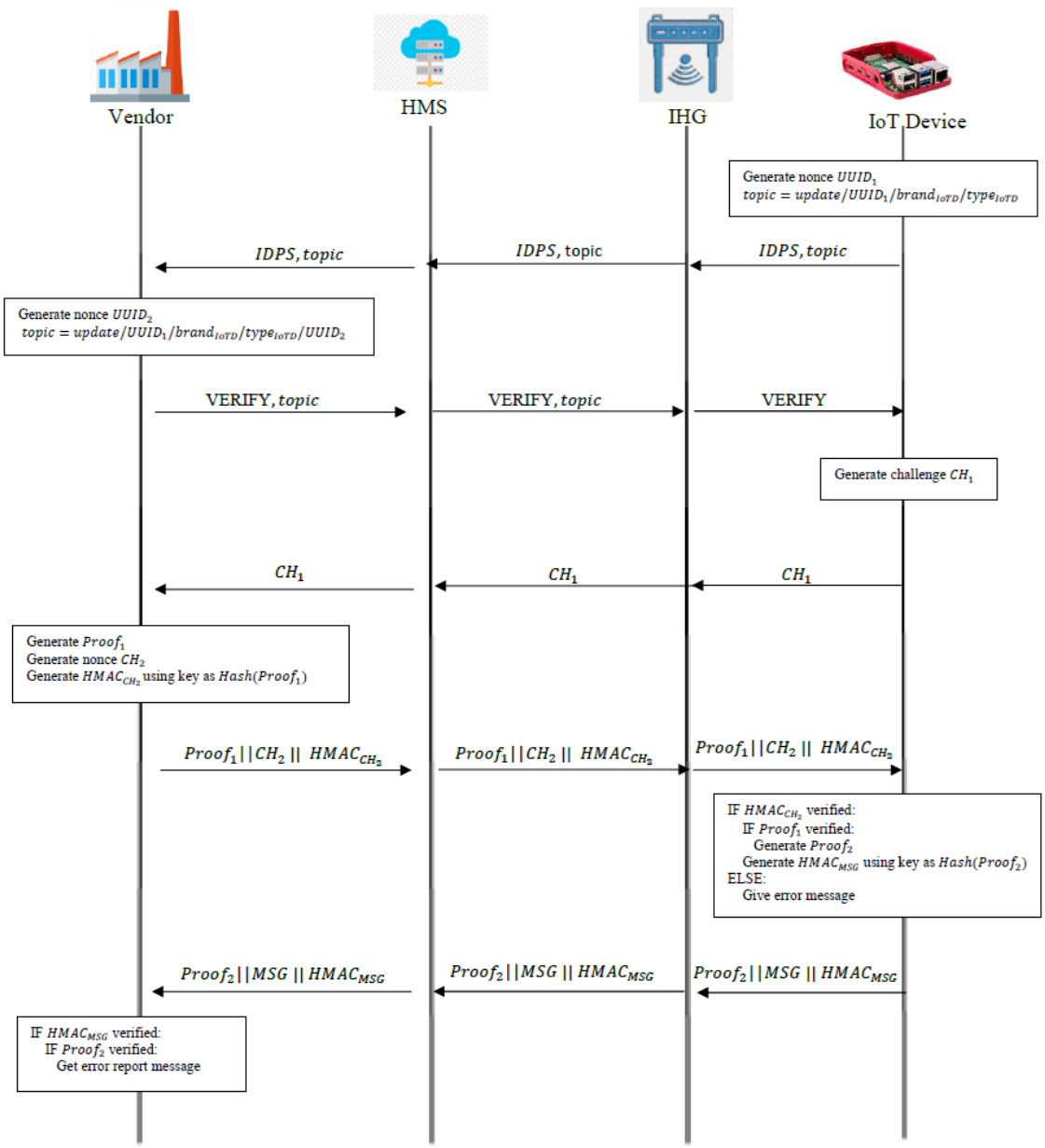

Figure 3.8 Mutual Authentication Protocol for Upstream Traffic Scenario

3.4.2.2 Protocol Flow of Upstream Traffic Scenario

Besides, the protocol flow for sending error reports to Vendor which corresponds to the upstream traffic scenario is almost the opposite of the explained downstream traffic scenario. However, there is a slight difference which is the creation of fake proofs. Fake proofs are only implemented for downstream traffic scenario in order to hide the real identities of IoT devices by proceeding communication with all of the connected devices even if they are not the target. On the other hand, in the upstream traffic error reports are sent to a specific Vendor. So, fake proof implementation is not performed.

For the sake of completeness, the protocol flow of upstream traffic scenario is given as follows:

- At first, Vendor is subscribed to the topic $report/+/brand_{IoTD}/+$ on which an $IoTD$ can publish the first message. $brand_{IoTD}$ is the brand of $IoTD$ as well as the name of the Vendor, so Vendor only gets the messages that are addressed to itself.

- The mutual authentication protocol is started by $IoTD$ by choosing the PS which includes the attributes of Vendor. Then $IDPS$ is sent to IHG over a TCP connection. $IoTD$ also generates a random $UUID_1$ to construct the specific topic and send it to IHG.

- After receiving the message, IHG publishes $IDPS$ to the topic $report/UUID_1/brand_{IoTD}/type_{IoTD}/$. As explained before, the role of IHG during the protocol is to relay the incoming messages to the other party in the communication.

- Vendor gets the message and generates another random $UUID_2$ to create a unique topic for the communication. Then, on topic $report/UUID_1/brand_{IoTD}/type_{IoTD}/UUID_2$, $VERIFY$ command is published by Vendor. From this point on, all the messages are published on that mutually constructed topic.

- IHG sends the $VERIFY$ command to $IoTD$ and $IoTD$ creates a random challenge CH_1 to start the authentication process.

- When Vendor receives CH_1, it generates $Proof_1$ to identify itself. Also, CH_2 is generated and HMAC of the challenge $HMAC_{CH_2}$ is calculated by using the key as $Proof_1$. Then, CH_2, $Proof_1$ and $HMAC_{CH_2}$ are published on the specific topic.

- Messages are delivered to $IoTD$ by IHG. First, $IoTD$ checks the validity of $HMAC_{CH_2}$ and if it is valid, it verifies $Proof_1$ in order to authenticate Vendor. If any of the verification processes fails, the execution stops. Otherwise, $IoTD$ generates $Proof_2$ using CH_2 and also it calculates $HMAC_{MSG}$ where MSG is the error report to be sent. After that, $Proof_2$, MSG and $HMAC_{MSG}$ are sent to Vendor over IHG.

- Lastly, Vendor checks the received $HMAC_{MSG}$ to validate the integrity of the error message. If it is valid, then the second verification is performed for $Proof_2$. After both verification processes are successfully passed, then Vendor installs the error report message (MSG).

4. EXPERIMENTAL RESULTS

In this section, we give experimental results on our prototype implementations. Performance results are discussed in several use case scenarios. Prototype implementations are performed based on the proposed architecture. Performance results are measured as the latency values of different scenarios. Experiments are conducted both for upstream and downstream traffic. For downstream traffic, we evaluate the performance results of sending updates from Vendor to IoT devices. On the other side, sending error reports from IoT devices to Vendor is the evaluated scenario for upstream traffic.

4.1 Setup and Implementation Details

Most of the development and testing process for IoT applications are supported with Raspberry Pi [29] which is a single board computer. So, for the implementation and performance evaluation phase, we used Raspberry Pi 3 and Raspberry Pi 4 as IoT devices in the system and a Raspberry Pi 4 with 4GB RAM works as IHG which is also customized as an access point for Wi-Fi connectivity. There exist 11 IoT devices in the testbed while 5 of them are Raspberry Pi 3 with 1GB RAM and 6 of them are Raspberry Pi 4 with 2GB RAM. The device which works as IHG is connected to the IoT network which is provided by the university via ethernet cable. All the IoT devices are connected to the access point via WLAN and to IHG via socket programming, other than that they are closed to any communication. The applications developed for the simulation are IoT devices, Vendor and IHG. They are implemented in IntelliJ IDEA with Java version 11. MQTT and Idemix libraries are attached to the applications as jar files so that the functionalities of the libraries can be used. Public parameters that are needed during the protocol simulation are served on a publicly accessible server.

Moreover, developed applications are built as jar executables so that it is more feasible to execute via Command Line on any platform. Jar executable of the IoT device application runs on all of the Raspberry Pi devices while the jar executable of IHG runs on Raspberry Pi 4 which is customized as an access point. Mosquitto (MQTT server) [23] serves on a laptop with operating system as Windows 10 and jar executable of the application for Vendor runs on a laptop the operating system of which is MacOS Big Sur.

4.2 Performance Evaluation

To evaluate the performance of the proposed system, we measure end-to-end latency values of several scenarios. Experiments are performed fifty times for update and error report scenarios and average latency results are presented with respect to the number of IoT devices. This way, we eliminate the edge cases and get a more smooth outcome. In each case, average latency is calculated from the beginning of challenge creation to the end of the mutual authentication protocol. The payload data size that is sent at the end of the protocol is set to 10KB for outcomes to be uniform.

4.2.1 Update Scenario

In this section, we give experimental results of downstream traffic which is sending updates from Vendor to IoT devices. Update scenario is evaluated in three different cases: successful delivery of updates, failed delivery of updates and mixed (successful and failed) delivery of updates. First, we discuss the results for the successful update scenario where all the connected devices get updates addressed to them successfully. Then, we present the failed update scenario results where all the devices get failed updates and complete the protocol by generating fake proofs. Lastly, we demonstrate mixed update scenario results where some of the IoT devices are delivered successful updates while the rest of the devices in the testbed receive the failed updates.

4.2.1.1 Successful update

In this section, experimental results for successful delivery of updates are presented. The payload data size of an update file is fixed to 10KB. Table 4.1 demonstrates the breakdown of average computational delay (in milliseconds) at Vendor's and IoT devices' side, respectively. It is seen that total computational delay is approximately 1233 milliseconds. Overall end to end latency for one device receiving update is calculated as 1614 milliseconds where time spent for network activities is 381 milliseconds.

Table 4.1 Average computational delay results for successful update scenario

Calculation	Computational Delay (ms)	Device
creating challenge	0.48	Vendor
verifying HMAC	3.26	Vendor
verifying Proof	21.94	Vendor
creating Proof	16.52	Vendor
calculating HMAC	0.62	Vendor
creating Proof	770.02	IoT device
calculating HMAC	3.14	IoT device
verifying HMAC	4.1	IoT device
verifying Proof	412.7	IoT device
total computational delay	1232.78	

Also, the experiments are conducted in a way that multiple IoT devices are in communication with the Vendor and end up receiving updates successfully. In Table 4.2, overall end to end latency results can be found for 1-to-11 IoT devices. Additionally, Figure 4.1 shows average latency results of the experiments together with a linear trendline.

As expected, there is a linear increase in end to end latency while the number of IoT devices increases. Average time to receive updates successfully is in between approx. 1.6 seconds to 2.3 seconds. As seen in figure, the slope of increase is 0.06 second per device indicating the scalability of the proposed model.

Table 4.2 Average end to end latency results for successful update scenario

Number of IoT device	End to end latency (ms)
1	1614
2	1744
3	1796
4	1922
5	1958
6	2007
7	2073
8	2110
9	2144
10	2185
11	2321

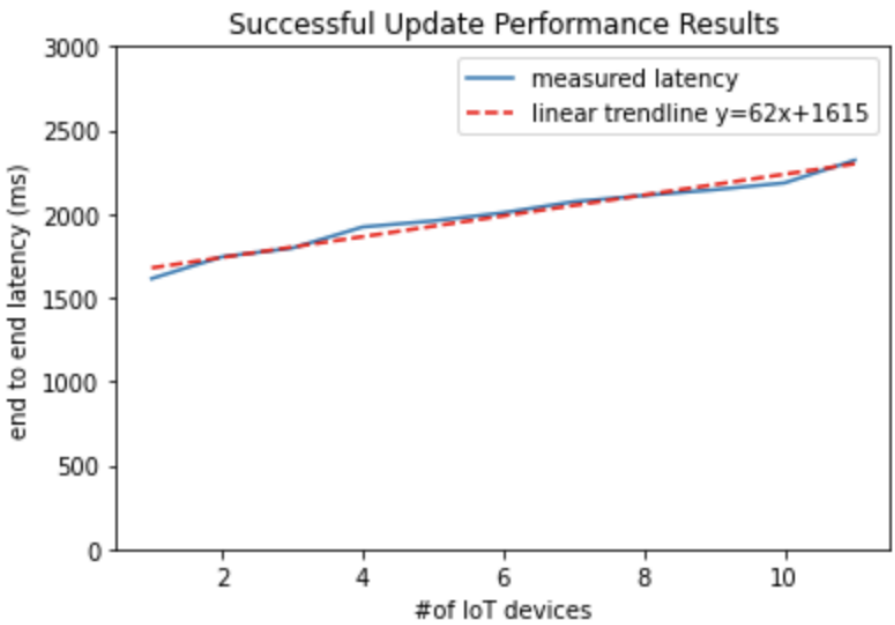

Figure 4.1 Average end to end latency for successfully receiving updates

Another experiment conducted for successful update scenario is measuring the latency values for different file sizes (payload data) to discuss applicability of the proposed scheme. For the uniformity of the outcomes, we fix the number of IoT devices to five. The experiments are conducted with 1KB, 10KB, 50KB and 100KB payload data sizes for update files and end to end latency values are presented in Table 4.3.

Table 4.3 Average end to end latency based on different file sizes for successful update

File size	End to end latency (ms)
1KB	1875
10KB	1958
50KB	2235
100KB	2728

One conclusion can be drawn from Table 4.3 is even when the file size increases from 10KB to 100KB, the increase in end to end latency is from approximately 1.9 seconds to 2.7 seconds which is linear and shows the scalability of the protocol. The difference between latency values is based on the computational delay of file read and write processes.

4.2.1.2 Failed update

As explained in Chapter 3, privacy-awareness of the proposed protocol comes from the failed update scenario where the real identities of the IoT devices are hidden so that outsiders cannot realize the actual device to which the update is addressed. As in the successful update case, the payload data size of an update file is fixed to 10KB. However, in failed update scenario, at the end of the protocol the update file is received but not installed on IoT device's side since the authentication fails.

Average computational delay breakdown for one device receiving a failed update can be seen in Table 4.4. Total computational delay is calculated as approximately 395 milliseconds where average end to end latency is 851 milliseconds for one device. As a result, communication delay is 456 milliseconds. It is expected that computational delay for failed update scenario is much smaller than the successful update scenario since generating and verifying fakeProofs are faster than legitimate Proofs.

Table 4.4 Average computational delay results for failed update scenario

Calculation	Computational Delay (ms)	Device
creating challenge	0.44	Vendor
verifying HMAC	3.22	Vendor
verifying fakeProof	10	Vendor
creating fakeProof	0.7	Vendor
calculating HMAC	0.62	Vendor
creating fakeProof	360.28	IoT device
calculating HMAC	3.1	IoT device
verifying HMAC	4.88	IoT device
verifying fakeProof	6.98	IoT device
total computational delay	394.86	

For the sake of comparisons, we conducted experiments where multiple IoT devices have received failed updates simultaneously. Average end to end latency values for multiple IoT devices are shown in Table 4.5. The experimental results for failed update scenarios are in the range of approximately 0.85 seconds to 1.5 seconds and the increase is linear with respect to the number of devices.

Table 4.5 Average end to end latency results for failed update scenario

Number of IoT device	End to end latency (ms)
1	851
2	895
3	937
4	1045
5	1075
6	1118
7	1173
8	1252
9	1298
10	1386
11	1461

Also, Figure 4.2 shows the linear trendline of the average end to end latency values for 1-to-11 IoT devices. As the figure indicates, the increase is 0.06 second per device slope. We can conclude that the proposed scheme shows good scalability for failed update scenarios.

To compare with, in Figure 4.1 and Figure 4.2 it is seen that successful update scenarios and failed update scenarios have the similar trendline which indicates that similar applications provide consistency.

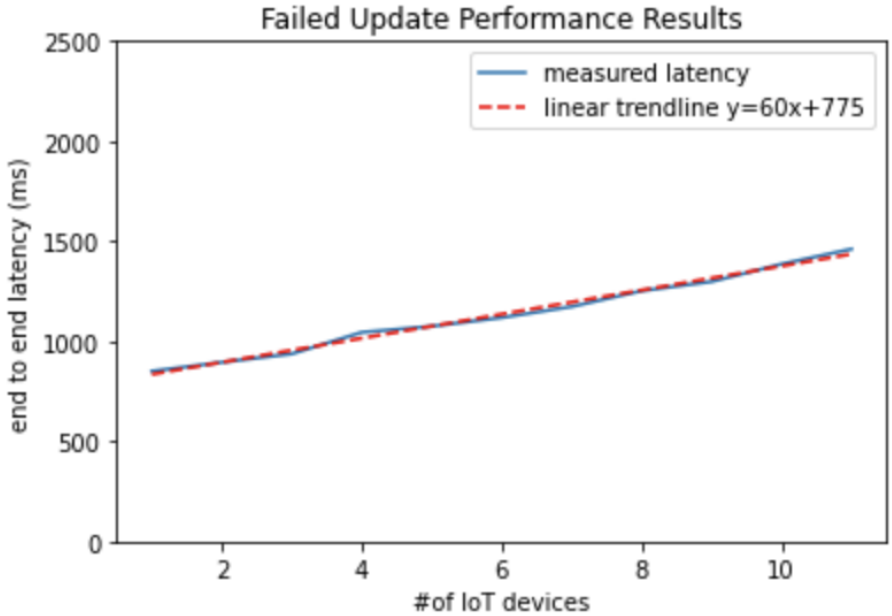

Figure 4.2 Average end to end latency for receiving failed updates

Moreover, we conduct experiments with different payload data sizes for update files. In the experiments, the testbed includes five IoT devices for the sake of uniformity. In Table 4.6, end to end latency values with respect to different file sizes can be found. While file size increases from 10KB to 100KB, overall latency increases from 1075 milliseconds to 1540 milliseconds. The increase in latency is expectedly linear and depends on the file read/write processes at the end of the protocol.

Table 4.6 Average end to end latency based on different file sizes for failed update

File size	End to end latency (ms)
1KB	859
10KB	1075
50KB	1244
100KB	1540

When the results for 10KB and 100KB file sizes are compared, the failed update scenario has less increase in latency in comparison to the successful update scenario. This is because, in the failed update case, the update file is only sent from the Vendor but not downloaded on the IoT device's side since the device is not the target for the update.

4.2.1.3 Mixed update

In this section, we present experimental results of mixed and simultaneous updates where some of the devices in the testbed receive successful updates and others fail to get updates. In Table 4.7, average end to end latency results of 11 IoT devices in such mixed scenarios are presented with respect to the number of successful updates.

Table 4.7 Average end to end latency results for mixed update scenario

Number of successful/failed updates	End to end latency (ms)
1 successful / 10 failed	1707
2 successful / 9 failed	1897
3 successful / 8 failed	1950
4 successful / 7 failed	2001
5 successful / 6 failed	2099
6 successful / 5 failed	2120
7 successful / 4 failed	2123
8 successful / 3 failed	2135
9 successful / 2 failed	2186
10 successful / 1 failed	2224

It is seen that for one device receiving the update successfully and ten devices failing to get update, the overall latency is 1707 milliseconds which is higher than the result of the failed update scenario with eleven IoT devices. This is because even just for one successful delivery of the update, the legitimate Proofs are generated and the mutual authentication protocol is ensured. Still, it can be concluded that the increase with respect to the number of successful updates is linear. Figure 4.3 demonstrates the linear trendline for mixed update scenarios.

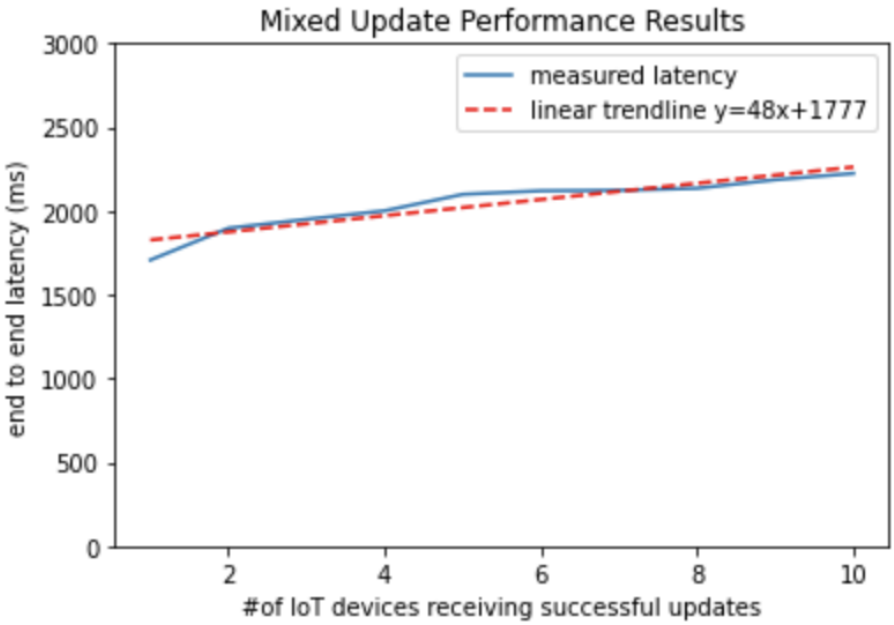

Figure 4.3 Average end to end latency for mixed updates simultaneously

As shown in Figure 4.3, the overall latency values are close to each other. The failed communications do not make a big difference in terms of end to end latency since at least one successful update is received by an IoT device in the testbed. The slope of increase in Figure 4.3 is approximately 0.05 seconds per device that receives a successful update.

4.2.2 Error Report Scenario

In this section, we present the experimental results for successful delivery of the error reports which corresponds to the upstream traffic scenario. The communication starts with the request of an IoT device indicating that the error report will be sent. After completion of the mutual authentication protocol, an error report which is assumed to have 10KB payload data size is sent to Vendor.

In Table 4.8, breakdown of average computational costs for one device are given. Total computational time is calculated as 1211 milliseconds and overall end to end latency for one device is 1337 milliseconds. Communication cost is calculated as 126 milliseconds.

Table 4.8 Average computational delay results for error report scenario

Calculation	Computational Delay (ms)	Device
creating Proof	0	Vendor
calculating HMAC	0.4	Vendor
verifying HMAC	0.64	Vendor
verifying Proof	23.16	Vendor
creating challenge	1.28	IoT device
verifying HMAC	24.72	IoT device
verifying Proof	433.36	IoT device
creating Proof	723.32	IoT device
calculating HMAC	4.88	IoT device
total computational delay	1211.76	

Table 4.9 provides the overall end to end latency results for multiple IoT devices from 1-to-11. Similar to the update scenarios, the devices operate simultaneously. Also, Figure 4.4 demonstrates the linear increase in overall latency with a slope of 0.08 seconds per device.

Table 4.9 Average end to end latency results for error report scenario

Number of IoT device	End to end latency (ms)
1	1337
2	1590
3	1668
4	1749
5	1865
6	1897
7	1917
8	1967
9	2074
10	2145
11	2291

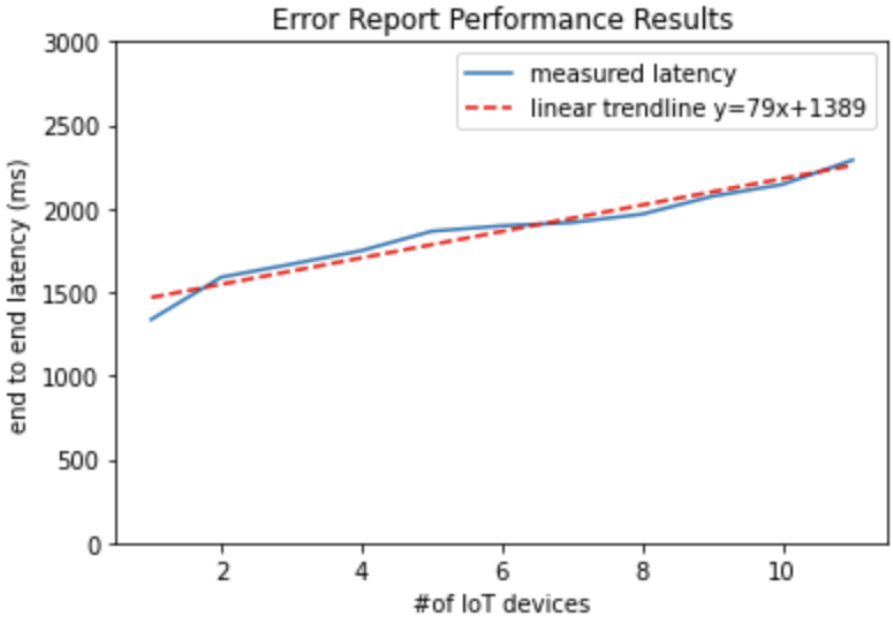

Figure 4.4 Average end to end latency for sending error report

4.3 Discussion

In this section, the experimental results are discussed and compared. To begin with, end to end latency in failed update scenarios is lower than the successful updates scenarios since the generation and verification of fakeProofs are much quicker than legitimate Proofs. In Table 4.1, it is seen that creating proof takes 770.02 milliseconds and verifying proof takes 412.7 milliseconds at the IoT device's side while Table 4.4 indicates that creating fakeProof takes 360.28 milliseconds and verifying fakeProof takes 6.98 milliseconds. To conclude, generating legitimate Proof takes almost twice as long as generating fakeProofs and verification process of fakeProof is too fast, even negligible. However, both scenarios have 0.06 seconds slope of increases confirming that the experiments give similar scalability performance under the same conditions.

On the other hand, slope of increase in error report scenarios is a bit higher than update scenarios since IoT devices perform most of the computations in error report scenarios. That's why it is expected to have a higher increase in latency per device. The communication time in error report and update scenarios are similar and considerably low indicating the feasibility of the proposed solution.

When the mixed update scenario results are examined, it is seen that the overall latency when an IoT device gets a failed update and ten IoT devices receive successful updates is 2224 milliseconds. On the other hand, the average end to end latency in the update scenario with 11 IoT devices is calculated as 2321 which is an expected outcome since in the mixed update scenario one device getting a failed update reduces the computational and communication delay on the Vendor's side. Moreover, a mixed scenario with one failed ten successful updates has higher latency than a successful update scenario with 10 devices. The reason is that one more device in the communication even with a failed update increases the time spent for network activities.

The increase in end to end latency values are linear in all scenarios, as shown in Figure 4.1, Figure 4.1 and Figure 4.4. It means that when the number of IoT devices are increased, the overall latency values also increase. However, the increase per device slope in each scenario is quite low which indicates that the proposed anonymous identification scheme is scalable for home automation systems.

5. SECURITY ANALYSIS

In this section, we provide security analysis on the proposed model using a security protocol animator by simulating different attack scenarios. Also, we discuss the boundaries of the model based on the knowledge that an attacker can obtain from the system.

5.1 Security Analysis

To analyze the proposed system, we use a security protocol animator, SPAN, for Avispa [30]. SPAN simulates the provided simulation and outputs a chart of execution. In the intruder mode, the tool acts as an attacker and comes up with different attacks on the simulated protocol. For the proposed mutual authentication protocol, we use SPAN to simulate and validate the authenticity of the protocol. The simulation of mutual authentication protocol is given in Figure 5.1. CAS+ language is used to write the simulation with an appropriate syntax. One can check the CAS+ manual [31] to see the details for CAS+ language. Additionally, written CAS+ specification for mutual authentication protocol is posted on Github repository [32].

The intruder simulation is tested for authentication of both of the parties in the protocol by the generated Proofs based on credentials (Cred1 and Cred2). Figure 5.2 shows one of the simulations where the intruder performs man-in-the-middle attack by impersonating Vendor and IHG at the same time. In the intruder simulation, Proof(Cred1, CH1) is represented with the notation as hash(const1, nonce3) where hash(.) represents any function (proof generation in this case), const1 is the Cred1 since it is constant and private to the users and nonce3 is the challenge. It is seen that when hash(const1, nonce3) is received by the intruder, it cannot be changed and is directly sent to the Vendor. Also, Vendor generates another Proof(Cred2, CH2) which is simulated as hash(const1, nonce4). Even if the intruder intervenes

in the protocol, since it does not have a legitimate credential and cannot generate a proof, the proof is not altered and sent to IHG as it is. In addition, HMAC (as hash(nonce4) in the simulation) is calculated for challenge and the update message, so it protects the integrity of the interchanged messages and if intruder changes any bit of the messages, the parties understand that they are not communicating with a legitimate entity and they stop the execution. As a result, the intruder cannot be authenticated by any of the parties in the simulation and the mutual authentication protocol is proven to be safe by OFMC [33] and ATSE [34] specifications.

The results of safety analysis are given in Figure 5.3 with ATSE mode and in Figure 5.4 with OFMC mode. One can readily replicate our results in the SPAN tool with the codes provided in the GitHub repository [32].

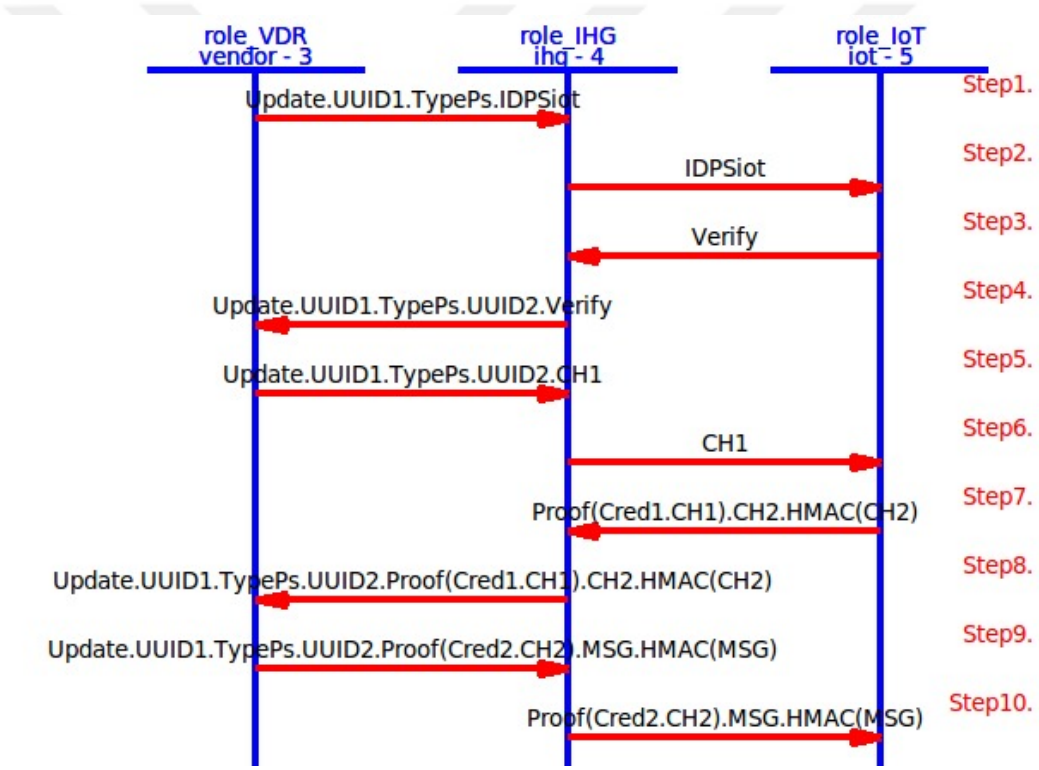

Figure 5.1 Mutual authentication protocol simulation using SPAN tool

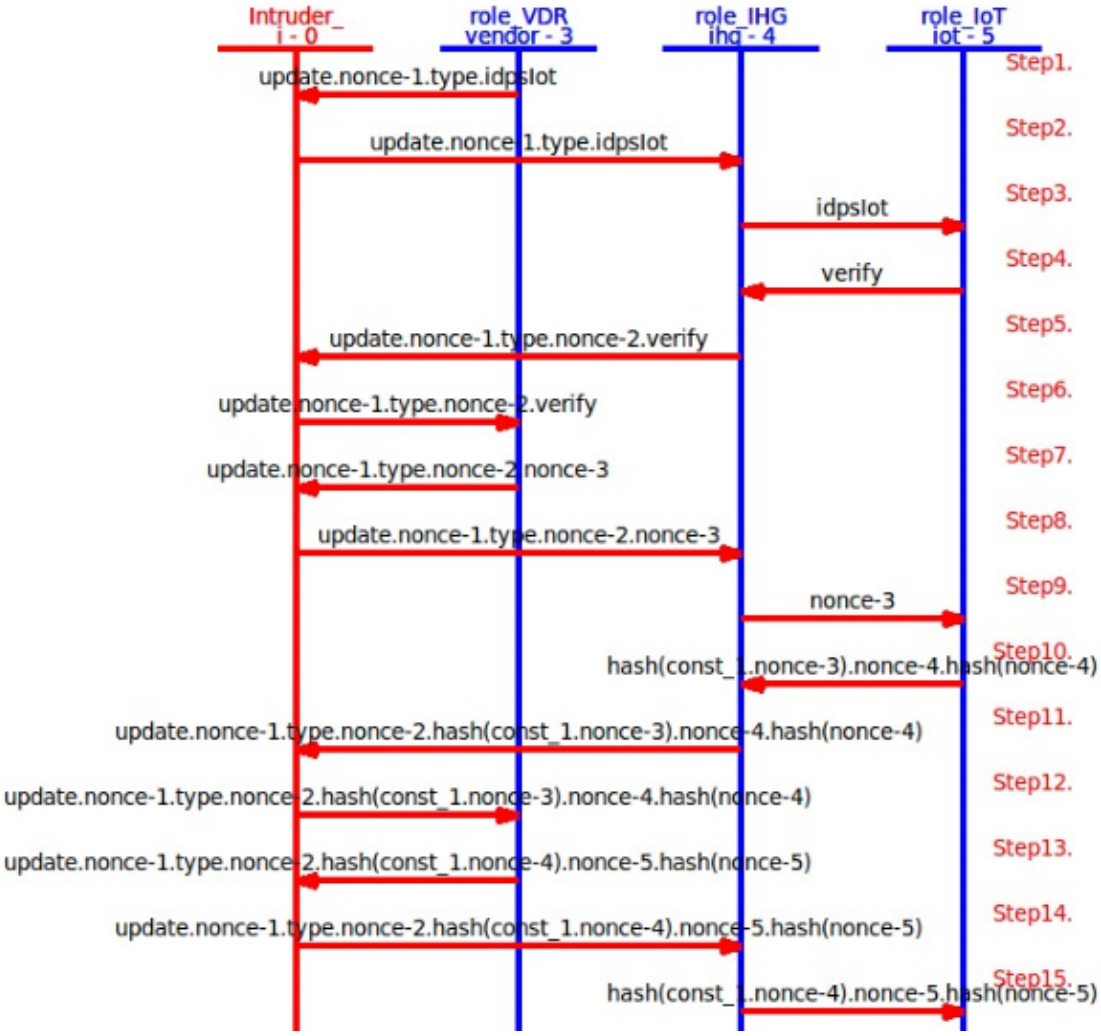

Figure 5.2 Mutual authentication protocol intruder simulation using SPAN tool

```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  /home/span/span/testsuite/results/hlpslGenFile.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS

  Analysed   : 14 states
  Reachable  : 9 states
  Translation: 0.04 seconds
```

Figure 5.3 Mutual authentication protocol safety analysis with ATSE mode

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/hlpslGenFile.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 1.12s
  visitedNodes: 753 nodes
  depth: 11 plies
```

Figure 5.4 Mutual authentication protocol safety analysis with OFCM mode

5.2 Discussion

In this section, we discuss the boundaries of the proposed model and suggest solutions for the potential weaknesses.

In the proposed model, fakeProofs are generated faster than real proofs. This may be a potential weakness in the system if all devices receive failed updates and the number of devices in the communication is known. If this information is leaked, then side-channel attack is possible by evaluating the overall latency to see whether the communication is completed too fast. We assume that the attacker cannot know the number of IoT devices at home so it can not realize the scenario where all of the devices receive failed updates. Still, it leaks information about attributes of the device so to avoid this information leakage, randomization can be implemented. This way, randomized data will be added to the communication and overall latency will be similar in different scenarios. In our model, to evaluate real-life latency values randomization is not implemented.

Also, encryption of the transferred data is not implemented. If the MQTT topic used in the communication is leaked then eavesdropping attack is possible since an attacker can subscribe to the topic and receive all the messages in the communication. Also, an attacker can learn device type and vendor name information by simply checking the topic name. This is another potential weakness in the system since the messages are transferred in plain text. However, we assume that the attacker cannot know the unique MQTT topic so the messages are sent over a safe channel. Still, to avoid this possible leakage encryption of the messages can be implemented as a future work.

6. CONCLUSION AND FUTURE WORK

In this thesis, we propose a privacy-aware Idemix based anonymization for home automation systems. The proposed system architecture includes IoT devices, Home Management System (HMS), Innovative Home Gateway (IHG) and Vendor. More precisely, IoT devices are connected to IHG which is a gateway responsible for relaying messages coming in to and going out from IoT devices. None of the entities in the system but IHG can directly communicate with IoT devices. To achieve anonymity, we adapted the Idemix anonymous credential system to our proposed model. By using anonymous credentials, we do not reveal the identity of the devices while we can make sure that the device is legitimate. To meet security requirements, we propose a privacy-aware mutual authentication protocol. In order for the communication to be completed, both entities need to authenticate themselves to the other party using their credentials. Moreover, the proposed mutual authentication protocol in which fakeProofs are presented is privacy-aware. With the introduction of fake proofs, we make sure that the communication is completed in an expected manner where no adversary can distinguish the failed communication. This way, the real identity of the IoT devices in the network is not revealed. Extensive experiments on the implementation of the scheme are presented and performance results are evaluated. Moreover, using SPAN protocol animator for the security analysis the proposed scheme is proven to be safe against man-in-the-middle attacks since the credentials are private to the users and adversaries can not authenticate themselves to the other parties without having a legitimate credential. Experimental results indicate that the proposed model is efficient and scalable for home automation systems.

As a future work, the proposed model can be extended in terms of preserving privacy of the users by obfuscating the communication between two parties. Also, for the secrecy of the interchanged messages, a secure key establishment protocol can be developed and symmetric encryption may be implemented.

BIBLIOGRAPHY

[1] W. H. Hassan *et al.*, "Current research on internet of things (iot) security: A survey," *Computer networks*, vol. 148, pp. 283–294, 2019.

[2] I. Lee and K. Lee, "The internet of things (iot): Applications, investments, and challenges for enterprises," *Business Horizons*, vol. 58, no. 4, pp. 431–440, 2015.

[3] Y. Upadhyay, A. Borole, and D. Dileepan, "Mqtt based secured home automation system," in *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*, pp. 1–4, IEEE, 2016.

[4] P. Kumar and U. C. Pati, "Iot based monitoring and control of appliances for smart home," in *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, pp. 1145–1150, IEEE, 2016.

[5] A. C. Jose and R. Malekian, "Smart home automation security: a literature review," *SmartCR*, vol. 5, no. 4, pp. 269–285, 2015.

[6] A. Gai, S. Azam, B. Shanmugam, M. Jonkman, and F. De Boer, "Categorisation of security threats for smart home appliances," in *2018 International Conference on Computer Communication and Informatics (ICCCI)*, pp. 1–5, IEEE, 2018.

[7] M. Weber and M. Boban, "Security challenges of the internet of things," in *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 638–643, IEEE, 2016.

[8] J. M. Batalla and F. Gonciarz, "Deployment of smart home management system at the edge: mechanisms and protocols," *Neural Computing and Applications*, vol. 31, no. 5, pp. 1301–1315, 2019.

[9] "Mqtt: The standard for iot messaging." Retrieved April 21, 2021, from https://mqtt.org/.

[10] J. Camenisch and E. Van Herreweghen, "Design and implementation of the idemix anonymous credential system," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 21–30, 2002.

[11] J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols," in *International Conference on Security in Communication Networks*, pp. 268–289, Springer, 2002.

[12] J. Liu, Y. Xiao, and C. P. Chen, "Internet of things' authentication and access control," *International Journal of Security and Networks*, vol. 7, no. 4, pp. 228–241, 2012.

[13] A. Alcaide, E. Palomar, J. Montero-Castillo, and A. Ribagorda, "Anonymous authentication for privacy-preserving iot target-driven applications," *Computers & Security*, vol. 37, pp. 111–123, 2013.

[14] X.-J. Lin, L. Sun, and H. Qu, "Insecurity of an anonymous authentication for privacy-preserving iot target-driven applications," *Computers & Security*, vol. 48, pp. 142–149, 2015.

[15] Z. A. Alizai, N. F. Tareen, and I. Jadoon, "Improved iot device authentication scheme using device capability and digital signatures," in *2018 International Conference on Applied and Engineering Mathematics (ICAEM)*, pp. 1–5, IEEE, 2018.

[16] Y. Zhang, R. H. Deng, G. Han, and D. Zheng, "Secure smart health with privacy-aware aggregate authentication and access control in internet of things," *Journal of Network and Computer Applications*, vol. 123, pp. 89–100, 2018.

[17] L. Zhang, Y. Ye, and Y. Mu, "Multiauthority access control with anonymous authentication for personal health record," *IEEE Internet of Things Journal*, vol. 8, no. 1, pp. 156–167, 2020.

[18] L. Zhou, X. Li, K.-H. Yeh, C. Su, and W. Chiu, "Lightweight iot-based authentication scheme in cloud computing circumstance," *Future Generation Computer Systems*, vol. 91, pp. 244–251, 2019.

[19] F. Li and B. McMillin, "A survey on zero-knowledge proofs," in *Advances in Computers*, vol. 94, pp. 25–69, Elsevier, 2014.

[20] J. Camenisch *et al.*, "Specification of the identity mixer cryptographic library," *IBM Research—Zurich*, 2013.

[21] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," *Communications of the ACM*, vol. 28, no. 10, pp. 1030–1044, 1985.

[22] D. Chaum and J.-H. Evertse, "A secure and privacy-protecting protocol for transmitting personal information between organizations," in *Conference on the Theory and Application of Cryptographic Techniques*, pp. 118–167, Springer, 1986.

[23] "Eclipse mosquitto." Retrieved June 11, 2021, from https://mosquitto.org/.

[24] "Mosquitto manual." Retrieved June 11, 2021, from https://mosquitto.org/man/mqtt-7.html.

[25] P. J. Leach, R. Salz, and M. H. Mealling, "A universally unique identifier (uuid) urn namespace," RFC 4122, July 2005.

[26] NIST, "Secure hash standard," Federal Information Processing Standard (FIPS) 180-4, 2012.

[27] H. Krawczyk, M. Bellare, and R. Canetti, "Hmac: Keyed-hashing for message authentication," RFC 2104, February 1997.

[28] S. Gur, S. Demir, S. Simsek, and A. Levi, "Secure and privacy-aware gateway for home automation systems," in *13th International Conference on Security of Information and Networks*, pp. 1–10, 2020.

[29] "Raspberry pi foundation." Retrieved May 30, 2019, from https://www.raspberrypi.org.

[30] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, *et al.*, "The avispa tool for the automated validation of internet security protocols and applications," in *International conference on computer aided verification*, pp. 281–285, Springer, 2005.

[31] T. G. Ronan Saillard, "Cas+ manual," 2011. Retrieved May 23, 2021, from http://people.irisa.fr/Thomas.Genet/span/CAS_manual.pdf.

[32] S. Demir, "Span, github repository," 2021. https://github.com/simgedemir/SPAN.git.

[33] S. Mödersheim and L. Vigano, "The open-source fixed-point model checker for symbolic analysis of security protocols," in *Foundations of Security Analysis and Design V*, pp. 166–194, Springer, 2009.

[34] M. Turuani, "The cl-atse protocol analyser," in *International Conference on Rewriting Techniques and Applications*, pp. 277–286, Springer, 2006.