Privacy Preserving Middleware Platform for IoT

Patrícia Raquel Vieira Sousa
Programa Doutoral em Informática das Universidades do Minho, Aveiro e Porto
Departamento de Ciência de Computadores
Ano

Orientador
Luís Filipe Coelho Antunes, Professor Catedrático
Faculdade de Ciências da Universidade do Porto

Coorientador
Rolando da Silva Martins, Professor Auxiliar
Faculdade de Ciências da Universidade do Porto

To my parents and João, for their love and support.

Acknowledgments

"Study while others are sleeping; work while others are loafing; prepare while others are playing; and dream while others are wishing."

— William A. Ward

This thesis had a duration of four years where many people were involved, directly and indirectly.

First, I would like to thank my supervisors, Professor Luís Antunes and Professor Rolando Martins, for all the support, guidance, and encouragement they have provided me since I started this Ph.D., and also for the opportunity to develop this work.

To all my colleagues in C3P: João Resende, Ana Carvalho, André Cirne, Inês Martins, Simão Silva and Luís Magalhães, not only for the great work/contributions we made during this period, but also for the fun times, conversations and... coffee breaks (... at least, before the COVID-19 pandemic lockdown ...). A special word for João Resende and Ana Carvalho, who share the office with me, for all their support during this period, even in the darkest moments.

To my friends, especially Joana Martins, for all the support not only during this thesis, but throughout my life.

I would also like to thank my parents Rogério and Teresa, for all the love and support. Without their confidence and faithful support, this journey would not have been possible.

Finally, last but not least, I want to thank my soul-mate João for all kindness, comprehension, patience, friendship, love, and tremendously patient on the best and difficult times. His encouragement and support helped me to overcome the most darkest hours and challenges.

This work was supported by Fundação para a Ciência e Tecnologia (FCT), Portugal (SFRH/BD/135696/2018). I am very grateful to the Fundação para a Ciência e Tecnologia (FCT), which made this dissertation and all my research possible by financing me directly with a Ph.D. scholarship. I am also grateful to CRACS and C3P, the two research laboratories where I had worked during my PhD, for receiving me and giving me the best conditions for research.

Abstract

The growing deployment of Internet of Things (IoT) technologies and the different ways sensors are connected, acquire and use personal data highlight the need for transparency, control, and tools to ensure that users' privacy is met in increasingly complex configurations. Due to the great heterogeneity of devices and communications in these systems, users' data becomes vulnerable and exposed. The IoT brings us a reality from a ubiquitous computational perspective, where data is shared on the Internet without users' control. Most of the time, the information exchanged contains confidential and private data about consumers or companies. The current assurances of IoT manufacturers do not meet current and potential consumers' growth expectations. Recent literature has highlighted some significant barriers to the growth of IoT, such as identity management, data protection technologies, data ownership, and privacy-preserving frameworks.

This thesis focuses on a solution to give users control over their data. We present an architectural design and implementation of two main modules: (a) a middleware layer to control all data shared with the Internet and (b) a secure provisioning module integrated with the middleware for end-to-end authentication between devices. This thesis innovates by positioning the users as active players in their data's control and market, behaving as data brokers for potential end-users data.

We started by identifying and reviewing privacy-preserving technologies, identity management, and end-to-end solutions, focusing mainly on IoT. Unlike existing provisioning methods, this thesis proposes a solution that gives the devices an identity, eliminating the risk of impersonating attacks and allowing devices to authenticate with each other. Finally, we integrate this solution with a middleware layer that gives the users the ability to control the privacy of all their data and is independent of the device's SDK, which can be deployed both on the home router and on the devices themselves so that it can be easily integrated into multiple use-case scenarios.

Resumo

Com o crescimento do conceito ”Internet of Things”, há cada vez mais dispositivos ligados à Internet a trocar informações entre si, com serviços online e com a cloud. Com este aumento da comunicação e transmissão de dados, aumenta também a necessidade de transparência, controlo e ferramentas que assegurem privacidade aos utilizadores, em relação aos seus dados. Devido à heterogeneidade dos dispositivos e comunicações, os dados dos utilizadores ficam mais expostos e vulneráveis.

Muitas vezes, a informação partilhada contém dados confidenciais e privados acerca dos próprios utilizadores ou empresas, e geralmente os utilizadores nem sabem que estão a partilhar esses dados. Por essa razão, é importante garantir que os fabricantes e fornecedores de produtos e serviços de IoT protejam os consumidores e a privacidade dos seus dados. As medidas atuais não correspondem ao grau de necessidade de privacidade que os utilizadores dos dispositivos IoT devem ter.

Os recentes avanços da literatura destacam algumas barreiras significativas para o crescimento da IoT, como a falta de serviços de descentralização, gestão de identidade, propriedade de dados e tecnologias de proteção de dados. Esta tese, foca-se na apresentação de uma solução que dá ao utilizador o controlo dos seus dados no IoT. Apresentamos um desenho de arquitetura e implementação com dois módulos principais: (a) uma camada de middleware que controla a partilha dos dados com a Internet e (b) um módulo de aprovisionamento de dispositivos, integrado com o middleware, que garante identidade e, consequentemente, comunicações autenticadas ponto-a-ponto entre os dispositivos. Esta tese inova porque posiciona o utilizador como um ativo no controlo dos seus dados e na comercialização, comportando-se como um broker dos seus próprios dados.

Nesta tese, começamos por rever o estado da arte das tecnologias de privacy-preserving, gestão de identidade e autenticação ponto-a-ponto, com foco no conceito de IoT. Ao contrário dos métodos de aprovisionamento existentes, esta tese apresenta uma solução que dá uma identidade aos dispositivos, eliminando a possibilidade de ataques de impersonating (onde o atacante faz-se passar por um utilizador fidedigno, de forma a aceder a informação priviligiada). Com esta solução, os dispositivos também conseguem autenticar-se mutuamente. Por fim, integramos esta solução com a camada de middleware que dá aos utilizadores a capacidade de controlar a privacidade de todos os seus dados e é independente do SDK do dispositivo, o que significa que pode ser implementado tanto no router doméstico como nos próprios dispositivos, para que possa ser facilmente integrado em casos de uso diferentes.

Contents

List of Tables

List of Figures

Glossary

AES Advanced Encryption Standard. 26, 54, 59, 63

API Application Programming Interface. 41, 46, 80–82, 84, 88–90, 92

ASN Autonomous System Number. 43, 94

BFT Byzantine Fault Tolerant. 19, 38

BLS Basic Life Support. 106

CA Certification Authority. 37, 53, 56, 64, 68, 73, 95

CBC Cipher Block Chaining. 59

CSR Certificate Signing Request. 70

DAG Directed Acyclic Graph. 21

DDoS Distributed Denial of Service. 45

DH Diffie-Hellman. 40, 41, 49–51, 64, 65, 69

DID Decentralized Identities. 38

DLT Distributed Ledger Technology. 38

DNS Domain Name System. 37, 43–45, 87, 89, 91, 92, 94

DoH DNS over HTTPS. 44, 87

DoT DNS over TLS. 44

DSA Digital Signature Algorithm. 69

EC Elliptic Curve. 74

ECC Elliptic Curves Cryptography. 42, 68, 69, 73, 74

ECDH Elliptic-Curve Diffie-Hellman. 68–71, 74

ECDSA Elliptic Curve Digital Signature Algorithm. 68–71, 74

ESNI Encrypted Server Name Indication. 44

FQDN Fully Qualified Domain Name. 43

GC Garbled Circuits. 22, 26–28

GDPR General Data Protection Regulation. 8, 35, 45, 84

GPIO General Purpose Input Outputs. 58, 93

HE Homomorphic Encryption. 25

HSM Hardware Security Module. 68

HTTPS HyperText Transfer Protocol Secure. 43, 44, 95

IAM Identity and Access Management. 34, 102

IBE Identity Based Encryption. 42

ICU Intensive Care Unit. 105

IdM Identity Management. xv, 10, 12, 32–38, 97–99, 101–105, 110–112, 114

IdP Identity Provider. 33, 34

IEFT Internet Engineering Task Force. 44

IFC Information Flow Control. 116

IMEI International Mobile Equipment Identity. 35

IoT Internet of Things. v, vii, 7–13, 15–21, 28–32, 34–38, 40–42, 44–46, 49, 50, 53, 58, 59, 65, 67, 68, 70–72, 76, 77, 79–81, 83–90, 93, 95–99, 101–105, 107, 108, 110–112, 114–116

IP Internet Protocol. 8, 37, 42, 43, 79, 87, 94

IR Infrared. 58, 60

ISP Internet Service Provider. 43, 44, 46

J-PAKE Password Authenticated Key Exchange by Juggling. 41

KC Key Continuity. 52, 55

LFD Large Format Displays. 22

LLC Location-Limited Channel. 42, 57, 60, 63, 66, 114

MAC Media Access Control. 87, 91

MANET Mobile Ad-hoc Network. 33

MiTM Man-in-the-Middle. 10, 28, 39, 41, 43, 49, 51, 55, 57, 63, 65–67, 78, 95, 99, 100, 114

MPC Multi-Party Computation. 10, 12, 15, 16, 22–31, 50–57, 65, 114

NFC Near Field Communication. 60

OCSP Online Certificate Status Protocol. 56, 57

OT Oblivious Transfer. 23, 26

OTP One Time Password. 10, 40, 42, 67, 68, 70, 73–77, 86, 114

P2P Peer-to-Peer. 31, 36

PAKE Password-Authenticated Key Agreement. 41, 42

PGP Pretty Good Privacy. 39

PII Personally Identifiable Information. 116

PIN Personal Identification Number. 70, 74, 75, 77

PIV Personal Identity Verification. 73

PKCS Public Key Cryptography Standards. 68, 73

PKI Public Key Infrastructure. xvii, 33, 37, 39, 40, 49, 53, 56, 57, 60, 64, 68, 114

PoC Proof of Concept. 19, 21, 73, 86

PoL Proof of Luck. 19

PoS Proof of Stake. 21

PoT Proof of Trust. 19

PoW Proof of Work. 16–19, 21, 40

PRNG Pseudo-Random Number Generator. 50, 64

PUF Physically Unclonable Function. 35, 42

PUK Personal Unblocking Key. 77

QR Quick Response. 41, 85

RBAC Role-Based Access Control. 33

RFC Request For Comments. 50, 52, 63

RFID Radio-Frequency IDentification. 7, 77

RP3 Raspberry PI 3. 53, 56, 58, 73, 75, 80, 86, 87, 89–91, 93, 96

RSA Rivest-Shamir-Adleman. 68, 73

RTP Real-time Transport Protocol. 58, 65

RTT Round Trip Time. 53

SAML Security Assertion Markup Language. 33–35

SAS Short Authentication Strings. 12, 41, 42, 49–58, 60, 63, 65, 66

SDK Software Development Kit. 25, 89, 93, 96

SFE Secure Function Evaluation. 26

SGX Software Guard Extensions. 76, 117

SIP Session Initiation Protocol. 57, 114

SMP Security Management Provider. 46

SNI Server Name Indication. 43, 44

SP Service Provider. 33, 34

SPOF Single Point of Failure. 21, 33, 37, 49, 64, 67, 68, 73, 76, 99, 109

SRTP Secure Real-time Transport Protocol. 41, 59

SSL Secure Sockets Layer. 40, 43, 57, 100

SSO Single-Sign On. 33, 34

TCP Transmission Control Protocol. 58, 60

TEE Trusted Execution Environments. 19

TLS Transport Layer Security. 40, 41, 43, 44, 56, 57, 70, 96

UDP User Datagram Protocol. 58

UI User Interface. 45

UID Unique IDentifier. 34

USB Universal Serial Bus. 77, 86, 93

VLAN Virtual Local Area Network. 92

VoIP Voice Over Internet Protocol. 40, 49, 50, 63

VPN Virtual Private Network. 28

ZRTP Z and Real-time Transport Protocol. 12, 40, 49–54, 57, 59, 63, 65, 114

Chapter 1

Introduction

”Arguing that you do not care about the right to privacy because you have nothing to hide is no different than saying you do not care about free speech because you have nothing to say.”

— Edward Snowden

IoT is a technological concept where all the devices of our daily lives are connected to the Internet, such as clocks, appliances, or even clothes, acting in an intelligent and sensorial manner [1]. When we refer to ”things” in IoT, we describe any objects, sensors, software, and other technologies to connect and exchange data with other devices and systems, not necessarily using Internet connections. These information exchanges can occur through many communication protocols, such as Radio-Frequency IDentification (RFID), WiFi, Ethernet, and Bluetooth, among other forms of connection.

IoT technology has several applications, from the simplest ones, such as turning a lamp on and off through the smartphone, to larger and more complex systems, such as smart cities and end-to-end industrial processes. In addition to generating more comfort and reducing day-to-day obstacles, IoT can optimize tasks and reduce costs, such as house automation. Users can automate their houses with doors that automatically open when the owners’ car is arriving home, washing machines controlled by smartphones, touchscreen mirrors that show news and videos, air conditioners that adjusts the house’s temperature even before the owner arrives, curtains that open and close according to the sunlight, and many other items 100% connected between them. In brief, smart homes consist of a set of equipment connected to the network, often remotely controlled by smartphones or voice assistants.

Authenticated users can securely control these devices via their smartphone and receive information and alerts from them. Anecdotally, a fridge can warn its owner when the food is close to the due date and search for the best prices of the items (or users’ preferences). Another possible example is a thermostat that can adjust the temperature according to environmental conditions and send information to the owner's smartphone. It is also possible to make homes more secure with wireless security cameras, sensors, and smoke alarms.

Besides, smart devices can help older people be more independent because these types of smart homes may include audible warnings or voice-activated alert systems that can help automate some tasks.

Despite so much ease and benefits brought by the connection of the objects surrounding us, IoT is in constant improvements and growth. This increasing pervasiveness can directly lead to privacy invasion, data leakage, and eventual viruses that can cause digital systems problems. What is the cost of having all these smart devices connected to the Internet when considering fundamental human rights, e.g., privacy? Also, these devices can communicate with their owner and other devices while potentially harvesting data. In turn, this data can be monitored by routers/switches that can relay it as telemetry to their manufactures. This opens the issue addressed in this thesis: with the increase in privacy and security issues in IoT, this thesis aims to create a middleware solution that provides mechanisms to handle device identity management and authentication and allows users to control their data.

In this first chapter, the main problem is contextualized and defined, and the main motivation for this work and its contributions are described.

1.1 Context and Problem Definition

The adoption of IoT is growing exponentially, and, according to *National Public Radio & Edison Research* [2], homes in the United States have an average of 2.6 smart speakers, explaining why the number of smart speakers is much larger than the number of its owners. *Jingjing Ren et al.* [3] conducted a study with 81 different smart devices to examine data sharing activities. The authors found 72 of the 81 IoT devices share sensitive data, such as external Internet Protocol (IP) addresses, device specifications and settings, usage habits, and location data, with third parties utterly unrelated to the original manufacturer, such as advertisers. This type of sharing can violate the users' privacy because, for example, according to the General Data Protection Regulation (GDPR), external IP addresses must be considered personal data as it falls within the scope of *online identifiers* or, more precisely, *Personal Identifiable Information* which is data that can identify a user [4]. This lack of consumer awareness is rapidly raising concerns about data privacy [5] as, for example, people who buy a Smart TV have no idea if their data is being shared or sold with technology providers from third parties. Worse, the authors found that 30 out of 81 devices shared data as an unencrypted text file. These examples raise significant privacy concerns, as those who collect these data streams can infer sensitive information, such as users' identity, location, or behavior.

The highly paced technological development surrounding it is exposing several novel challenges, namely on privacy and security. Lack of privacy protection mechanisms mainly refers to weak authentication and confidentiality [6], which leads to problems related to user data privacy. It is essential to improve data transparency for users and detection of potential attacks [7]. Also, there are other challenges related to resource limitations, heterogeneous devices, interoperability of security protocols, single points of failure, hardware/firmware vulnerabilities, and reliable management and updates [8]. In particular, there is a need to adopt security solutions for IoT devices characteristics due to their intrinsic limitations [9] (e.g., battery life and memory space). Recent literature [10–12] highlighted that privacy concerns could be a significant barrier to the growth of IoT, highlighting security and privacy as a significant IoT research challenge.

The rapid adoption of these devices connected to the Internet raises concerns about their ability to pry, meaning an increasing variety of IoT applications also creates new security challenges. There are several scenarios of IoT application domains, namely Transportation and Logistics, Healthcare, Smart Environment, and Personal/Social [13]. The health domain is one of the most motivational application fields because patients expect certain private information to remain confidential, but there is a lack of adequate mechanisms to ensure personal and confidential information [14]. In terms of security, if these types of systems are hacked or fail, it can lead to catastrophic consequences [15].

1.2 Motivation and Main Contributions

Today's smart home environments do not allow users to see the data sharing life cycle. Many data are shared without the users' knowledge, especially data collected by the device manufacturers during their regular operation, such as smartwatches sharing heart rate data or fridges sharing the number of items available to the supermarket or manufacturer. This type of data sharing should start to worry users, as much of the information must remain private or, at least, give the user the possibility to decide that. It often happens and is entirely invisible to the user, and there are no configurable users' preferences.

The control of permissions and user feedback needs to be improved by IoT devices. For example, in smartphones, the permission mechanisms are already known to users. There have been improvements at the feedback level, namely in the iPhone, which has a new feature and a green indicator that appears at the top of the users' screen whenever an app is using the microphone or camera. If users find an application that connects the camera but do not need it, they can revoke the camera's permissions for that specific application [16].

However, there are no known settings for a regular user on IoT, and it is not always easy to configure the basics, and there is often no transparency about privacy policies or possible settings at that level. IoT presents a significantly broader set of scenarios where it needs to be more transparent to users. For example, in most personal assistants, microphones are always on, and users do not know this, nor can they configure these permissions.

Data owners must understand the complete picture of what will happen behind the scenes with their data. This thesis focuses on developing a new middleware platform that addresses identity management and authentication mechanisms to ensure secure communication and deployment of a secure IoT environment and provides users with ways to control their data sharing.

To the development of the final solution, we focused mainly on two components. The first component focuses on the provisioning of devices because devices must authenticate with each other to ensure that there are no impersonation attacks, which allow, for example, a hacker to impersonate a trusted device and access privileged information. For this, we designed a solution that guarantees an identity for each ”thing” enabling end-to-end communications. We achieve several benefits: device identity, scalability, offline cryptographic assets, and resistance to Man-in-the-Middle (MiTM). This provisioning solution innovate by having device islands, which in addition to authenticating multiple devices, can authenticate with other device islands (for example, different entities or departments, and even in different cities).

The other component consists of a middleware layer for the router or the device itself to block data transmissions to the Internet. This component allows all visualization and interaction with the data, allowing the definition of privacy policies. The architecture integrates the implementation of the provisioning solution with the middleware layer.

We aimed for a decentralized solution that gives users the possibility to configure their privacy preferences. The platform places the user as an active player in the data market, behaving as its own data broker for the potential data end-users.

The work carried out during the thesis led to the following main contributions:

i overview of existing techniques for privacy-preserving in edge computing/fog computing without a trusted third-party, namely Multi-Party Computation (MPC) and Blockchain, discussing current unsolved problems and possible future research directions;

ii describe the most relevant IdM systems focusing on privacy-preserving with or without Blockchain and evaluates them against ten selected features grouped into three categories: privacy, usability, and IoT;

iii explore some research questions that include the requirements to build a privacy-preserving IdM system for IoT, analyzing current solutions’ features and their applicability to healthcare scenarios;

iv design and implementation of a novel approach for provisioning of IoT devices, giving an identity to a ”thing”, reducing the risks of impersonating attacks and enabling the devices to authenticate;

v combination of One Time Password (OTP) with cryptographic algorithms for providing a secure provisioning protocol;

vi a prototype with Yubikeys used to provision devices with cryptographic assets offline;

vii a middleware to improve the privacy of users' data on the IoT, with implementation and evaluation, enabling users to store data to be controlled offline and to analyze current connections, discarding them according to the user's preferences, without delay, extensible to network communications;

viii present and evaluate all the implementations of each component that composes the middleware;

ix an integration of the middleware solution with a marketplace, providing users the way to monetize their data;

x present and discuss the main open challenges.

1.3 Bibliographic Note

Some of the work described in this thesis has already been published. The following list provides a reference to such publications and the chapter to which the contribution is related.

- Patrícia R. Sousa, et al. "The present and future of privacy-preserving computation in fog computing." Fog Computing in the Internet of Things. Springer, Cham, 2018. 51-69. **(Chapter 2)**
- Patrícia R. Sousa, et al. "pTASC: trustable autonomous secure communications." Proceedings of the 20th International Conference on Distributed Computing and Networking. 2019. **(Chapter 3)**
- Patrícia R. Sousa, et al. "Secure Provisioning for Achieving End-to-End Secure Communications." International Conference on Ad-Hoc Networks and Wireless. Springer, Cham, 2019. **(Chapter 4)**
- Patrícia R. Sousa, et al. "Internet of things security with multi-party computation (mpc)" EP3570575A1 - European Patent Office. **(Chapter 3)**
- Patrícia R. Sousa, et al. "The case for blockchain in IoT identity management." Journal of Enterprise Information Management (2020). **(Chapter 2 and 6)**
- Patrícia R. Sousa, et al. "Empowering Users Through a Privacy Middleware Watchdog." International Conference on Trust and Privacy in Digital Business. Springer, Cham, 2020. **(Chapter 5)**

1.4 Thesis Outline

The next chapters of the thesis are organized as follows:

Introduction
This chapter presents the context and definition of this thesis's problem, together with this work's motivations and contributions.

Literature Review
The second chapter presents a review of previous work. A discussion is provided about the present and future of privacy-preserving computing in Fog Computing, reviewing some of the most used concepts and application scenarios. Then, we present the literature review of the different concepts that we covered during this thesis, mainly identity management, device provisioning, and end-to-end communications. Finally, we present an overview of some middleware solutions developed for IoT, which help point out our approach's main differences.

The Case for Blockchain in IoT Identity Management
The third chapter addresses Blockchain as a solution for IoT Identity Management, answering three fundamental research questions concerning the requirements for building an IoT privacy-preserving IdM system, Blockchain's use to meet these requirements, and the critical aspects applying blockchain-based IdM systems to healthcare settings. We provide a full discussion and several ideas as challenges for future research.

End-to-End Secure Communications
Some solutions to address end-to-end communications are covered in chapter four, namely, with the use Z and Real-time Transport Protocol (ZRTP) as an end-to-end protocol for data communication. We use mechanisms, such as MPC and infrared to compare a Short Authentication Strings (SAS) automatically privately. We present an evaluation of the proposals and discussions for future research challenges.

Secure Provisioning for Achieving End-to-End Communications
The fifth chapter addresses the secure provisioning of devices by combining a public-key cryptography algorithm with an OTP inside a secure token - the secure token acts as an offline storage for the private keys. Device identity is guaranteed by physical access to the physical token. Our final goal was to provide a solution to achieve device identity, scalability, offline cryptographic assets, and resistance to the impersonation, replay, and MitM attacks.

Empowering Users Through a Privacy Middleware Watchdog
The sixth chapter provides the central core of this thesis, presenting an initial design and implementation to integrate different components to address privacy-preserving in the IoT, focusing on users and their data. We present solutions for data sharing network control, as well as data control locally. Then, an integration with the authentication component provided in Chapter 5 is covered. Also, we

provide some insights into future research directions on the topic.

Conclusions

The seventh and final chapter concludes the thesis. It summarizes the motivations and contributions provided by this work and future directions for research on the topic of IoT privacy-preserving middleware solutions.

Chapter 2

Literature Review

"Every success story is a tale of constant adaptation, revision and change."

— Richard Branson

In this chapter, the work described in this thesis is framed, presenting a review of the work related to privacy preservation techniques, identity management, authentication and secure communications, network traffic security solutions, and data privacy middleware systems. Such topics were extensively studied and analyzed, presenting the approaches most similar to our thesis's goal and comparing each one's benefits and disadvantages.

2.1 Privacy-Preserving Computation for IoT and Fog/Edge Computing

IoT represents a remarkable transformation in how our world has interacted over the years. The growing increase in data traffic and real-time processing has overloaded networks and cloud systems. Fog computing leverages to "push" data processing and storage services close to the devices that generate them. Fog computing [17] is one of the supreme examples of edge systems and is one facet of the overarching concept - the IoT. A relocation of computing, preferably closer to devices near the end-user, can describe this concept of fog/edge computing.

The ever-increasing pervasiveness of edge computing is creating challenges for users' privacy. The need to reduce time-to-market has lead companies to deploy edge computing systems without security and privacy by design. The exponential growth of the IoT data generated and communications by this type of system is chronically exacerbating the problem. Mechanisms are needed to ensure the data's privacy, especially in IoT communications.

MPC [18–20] poses a suitable option to offer the basic building block for the construction of decentralized privacy-preserving computational frameworks. MPC is a sub-field of cryptography to create methods for parties to jointly calculate a function on their inputs, keeping them private. In 2015, the Enigma [21] technology was created, focused on decentralized computing with privacy guarantee based on secure MPC, eliminating the need for a reliable third party. This model works parallel with blockchain technology that controls the network, manages access control, identities and functions as a tamper-proof event log.

This section focuses on exploring blockchain, and MPC approaches (and the combination of both) to create solutions to enhance privacy-preserving approaches for IoT. We explored current literature and discussed the integration of several different approaches. We first describe three significant concepts: blockchain, IoT/fog computing, and MPC. These concepts provide the necessary context and background for developing possible research paths and ideas.

The remainder section is structured as follows: Section 2.1.1 describes blockchain with some practical frameworks and applications and then describes which ones can impact IoT. We then move to investigate and describe current approaches that combine blockchain, IoT, and fog computing. Lastly, we explore MPC in the Section 2.1.2. As it is a concept that promotes privacy without a third party, Section 2.1.3 presents both MPC and blockchain and the applications of the concepts together. Furthermore, we offer an overview of MPC potential frameworks and assess the feasibility of their integration with other privacy concepts. We conclude by discussing current unsolved problems and possible future research directions. Lastly, we summarize the finding in the Section 2.1.4.

2.1.1 Blockchain

Blockchain is a distributed database that maintains a list of sorted records in an increasing sequence, where different transactions are recorded as a block data structure. Each block is linked to the previous with a cryptographic hash function, thereby forming a blockchain [22, 23]. Each block has unique, immutable, and irreversible data values, ensuring that they are not changed.

There are currently two types of blockchain technologies: permissionless (public chains) and permissioned (private chains). Anyone can become part of the network without requiring an identity (and associated trust model) in a permissionless blockchain. In contrast, in a permissioned blockchain, only trusted agents could write and possibly read those records [24], thus requiring a well-established identity and trust among the cooperating nodes.

Previous work has studied and evaluated private and public blockchain for a joint venture [25]. The authors claim that private blockchain improves scalability in terms of large numbers of clients and transactions. In a permissioned blockchain, it is unnecessary to compute intricate consensus mechanisms, such as Proof of Work (PoW), because participants are known and white-listed, and a trusted authority must initially authorize them. To improve throughput, the ordering of blocks requires a broadcast protocol that offers total order. Depending on the type of faults targeted, these protocols can protect against crash faults (such as Kafka [26] does) or more reliable assurances by providing fault-tolerance against byzantine (arbitrary) faults [27], making private blockchain more efficient.

Public blockchains are decentralized and independent of a central authority. It needs consensus mechanisms that refer to reaching a cooperative agreement (consensus) on the current state of the distributed ledger [28]. It facilitates the verification and validation of the information added, ensuring that only authentic transactions are written to the blockchain, achieving reliability, and establishing trust between unknown peers in a distributed computing environment.

In Bitcoin [*], for example, the mining process in PoW systems involves an immense use of energy and computational resources. Therefore, a miner's performance is calculated by the amount of computational power it has, usually called the hash power or hash rate. There are several mining nodes in different locations, and they compete to find the next valid block hash, and if successful, they are rewarded with new Bitcoins units.

In this context, mining power is distributed among several nodes worldwide, which means that a single entity does not control the hash rate. However, the hash rate may not be distributed evenly, and, therefore, when a single entity or organization manages to obtain more than 50% of all hash power, it means that it controls most of the network. This is known as the 51% attack. A 51% attack refers to an attack on a blockchain when a person or group of miners controls more than 50% of the network's computing power because it would have the same mining capacity as all other mining groups. Public blockchain protocols are vulnerable to attacks that can take advantage of the need for consensus. If miners can control 51% of nodes operating on the network, they can manipulate core rules and take control of the system, being able to attack the network and rewrite the recent blockchain history, sensor transactions, e.g., for name registrations, and steal cryptocurrency using double-spend attacks [32].

This attack's success is generally associated with the failure of other associated security mechanisms, namely the lack of robust identity management. A Sybil attack allows an attacker to subvert a network service's reputation system by creating many pseudonym identities and using them to gain disproportionately large influence. In the case of blockchain, Sybil attack can cause severe impact public/permissionless blockchain, in which an attacker can subvert the blockchain by creating a large number of fake user accounts and push legitimate entities in the minority [33].

2.1.1.1 Motivation for using Blockchain with IoT

Blockchain has some recommended characteristics for resource-limited devices in IoT environments [34]. It ensures decentralization, which means sensors can exchange data directly with each other rather than use a third-party system to establish digital trust, reducing implementation and op-

[*]Bitcoin is a digital currency and online payment system, also called digital cash. It works in a decentralized way that uses peer-to-peer to enable payments between parties without mutual trust. Bitcoins are digital coins issued and transferred by the Bitcoin network [29–31].

eration costs by eliminating intermediaries. The IoT sharing is enhanced with reliable and traceable information, where the data source can be identified [35]. Besides, through an immutable ledger, blockchain also guarantees non-manipulation, which means that, in the case of identity and access management systems, it guarantees that the device information is genuine and that its software and settings have not been tampered with or breached.

For this reason, blockchain-based identity and access management systems can be leveraged to strengthen IoT security [36] by storing identity, credentials, and digital rights. As described by *Ali Dorri et al.* [37], blockchain can keep private user's identities and offers scalability and robustness by using the resources from all participating nodes, and in the case of IoT, from all devices.

Blockchain also gives devices autonomy through smart contracts. It provides self-sufficient programs stored in a decentralized manner and executed autonomously when certain conditions of a business process are met [38, 39].

2.1.1.2 Limitations of integrating Blockchain with IoT

Although there are some advantages of integrating blockchain with IoT environments (see the section 2.1.1.1), its integration into IoT still requires further research. In this section, we want to analyze the benefits and disadvantages of this integration, describing some optimization solutions that have been proposed over the years.

Blockchain is computationally heavy and costly in terms of power consumption [40]. It must be adapted to be suitable for resource-limited IoT devices, and, due to the massive amount of data on IoT, blockchain must handle billions of transactions between IoT devices.

To ensure that network utilization is within a prescribed operating range, throughput management can provide self-scalability, and as the network grows, there is availability for more transactions on the public blockchain. Besides that, lightweight consensus and distributed trust [41] are also essential enhancements. The design of a new distributed trust method improves processing time to validate new blocks as it gradually decreases as they build trust with each other. It is also possible to enhance these optimizations with a tiered architecture that uses a private, centralized immutable ledger to reduce overhead and a decentralized public with high-end devices for greater confidence that does not lead to further transaction delays processing [42].

In blockchain, the resolution of complex mathematical puzzles requires a substantial amount of computational power. It is possible to eliminate the need to solve any puzzle before attaching a block to the blockchain (PoW). Recently, a decentralized privacy-preserving healthcare blockchain system for IoT [43] eliminates PoW to make it suitable for IoT. Blockchain transactions are public, and there is additional information about senders and recipients on the blockchain network. Thus, this work uses a ring signature scheme that preserves privacy to provide anonymity to users.

Another optimization for the computational power is through fog and edge computing. The idea is to relocate part of the computing power in the data center to network boundaries. In the blockchain concept, computing can be transferred to edge servers near miners [44] to overcome substantial CPU time and power consumption issues.

The combination of smart contracts with blockchain automates time-consuming workflows, achieving verifiable encryption and high cost and time savings in the process [45]. Blockchain offers a resilient distributed peer-to-peer system and the ability to interact with peers reliably and audibly. Smart contracts allow to automate complex multi-step processes.

Blockchain's current approaches focus on process optimization based on Byzantine Fault Tolerant (BFT) and propose *Bitcoin NG* [46] to support billions of devices without the need for additional features. However, BFT protocols often have problems with node scalability [47], which is significant for blockchain over IoT applications. In terms of scalability, for example, BFT is worse than Bitcoin because Bitcoin scales well for thousands of nodes, while BFT scales only for a few dozen nodes [48]. They have the opposite behavior in terms of complexity, as BFT has less complexity than Bitcoin. However, it is also impractical to integrate Bitcoin directly into IoT. Previous work integrates Bitcoin with IoT to provide a multi-layered blockchain-based method for sharing IoT user data with organizations and people [49]. The authors assume that IoT devices do not have enough resources to solve PoW because it requires very sophisticated hardware, and we cannot consider direct integration as it requires facing some resource, latency, bandwidth, and scalability challenges. The same work from the authors analyzed the Practical BFT performance as a consensus algorithm; however, they conclude that it can be a bottleneck in networks with many peers [50].

There are some proposals for redefining consensus algorithms to be more suitable for IoT integration, such as Proof of Luck (PoL) [51] which is a consensus mechanism that relies on Trusted Execution Environments (TEE) capabilities with a blockchain design Proof of Concept (PoC) to offer low latency transaction validation, reduced power consumption, and evenly distributed mining. There is also ongoing research on consensus protocols to achieve scalability [52], such as Proof of Trust (PoT) [53], for example, which attempts to address scalability problems, particularly associated with BFT-based algorithms, and avoids the low throughput and resource-intensive associated with PoW mining.

In brief, a standard solution to the limitations of the integration between blockchain and IoT does not yet exist. For example, there are no effective solutions to the 51% attack or a standard definition of a lightweight blockchain, yet [54].

There are some unsolved critical challenges to successfully integrate blockchain with IoT, namely:

- Mining is computationally intensive;
- Mining of blocks (PoW) is time-consuming and energy-consuming;
- Blockchain scales poorly as the number of nodes in the network increases;
- The underlying blockchain protocols create significant overhead traffic. [37]

Future research directions are related to these limitations, namely security and privacy, connectivity and scaling, energy consumption, resource allocation, blockchain standardization, and model optimization.

2.1.1.3 The Internet of Things, Fog Computing and Blockchain

New solutions are needed to address the limitations of the integration between IoT and blockchain, and fog computing is a prime candidate to be employed in this scenario. One of fog computing's main goals is to deal with the current limitations of public cloud computing when dealing with services and applications that require very low latency, local awareness, and mobility (including vehicular mobility).

Follows the presentation of approaches that explore the integration of between IoT, blockchain, and fog computing. We will describe their applications and how they address the challenges above.

IOTA
IOTA [55] is a new transactional settlement and data transfer layer for IoT, and part of the name "IOTA" emphasizes the importance conceded to the IoT. With the growth of the number of devices in the IoT, which can reach tens of billions of connected devices in the next decade, one of the primary needs is interoperability and resource sharing. For this, IOTA lets companies explore new business-to-business models by making every technological resource a possible service to be traded on an open market in real-time and without fees.

IOTA combines both fog and mist computing [*] into a new distributed computing solution. This technology combines smart sensors with built-in computational capabilities (mist computing) with nearby processing stations (fog computing). These new paradigms' main goal is to decrease the network latency to cloud servers that can be located far away from end-devices. Hence, the industry must rely on a free real-time, low-latency, and decentralized settlement system [57].

IOTA micro-transactions enable party A's sensor data to be processed by party B's processors in real-time. In return, Party B can use the iotas it gets compensated with to buy data from Party A or any other technological resource from another party in this symbiotic ecosystem.

In this new autonomous machine economy, IOTA works as its backbone. The main innovation behind IOTA is the Tangle, a novel new block-less distributed ledger that promises to be more scalable, lightweight, and, for the first time, makes it possible to transfer value without any fees, contrarily to blockchain. Also, consensus is no-longer decoupled but instead an intrinsic part of the system, leading to decentralized and self-regulating peer-to-peer networks [55]. The Tangle ledger can settle transactions with zero fees, contrarily to the traditional blockchain, so devices can trade exact amounts of resources on-demand and store data from sensors and data loggers securely and

[*]Mist computing decreases latency and increases subsystems' autonomy. [56]

verified on the ledger [55].

The relationship between IOTA and Tangle is similar to the relationship between Bitcoin and blockchain. One main differentiating factor is that IOTA does not use sequential blocks as the blockchain, but instead, it uses Tangle [58], a Directed Acyclic Graph (DAG) where each node graph is a transaction. Instead of having miners for validating transactions, the network participants must confirm two transactions already submitted to the network for every one transaction that they issue, being jointly responsible for transaction validation. As there is no miner, the transaction initiator does not need to pay the transaction service fee. This zero-service fee design is especially suitable for the data exchange in the future IoT era. However, although the transaction is free of charge, PoW is still required to prevent spamming, similar in spirit to the PoW used in blockchain. However, it is a straightforward computational operation because it is only used as a protection mechanism, having almost no impact on consensus.

In theory, Tangle supports micro-payments with a special focus on data integrity and precision.

IOTA claims [59] that it is faster and more efficient than typical blockchains used in cryptocurrencies.

ADEPT

The exponential growth that IoT is experiencing is making it increasingly essential to have decentralized networks as a mean to eliminate Single Point of Failure (SPOF) that are associated with traditional centralized networks, as a way to increase its robustness and reduce the infrastructure and maintenance costs to manufacturers and vendors. Using the devices themselves as computational, storage, and communication nodes, it is possible to construct ”hybrid” IoT systems where the ”edge” complements centralized systems. IBM and Samsung have developed the ADEPT PoC to establish a foundation on which to demonstrate several capabilities that are fundamental to building a decentralized IoT [60]. The authors demonstrated a distributed system capable of sustaining a fully decentralized framework for IoT. As its backbone, ADEPT uses the blockchain to build a decentralized and distributed network of things [61, 62], using a combination of PoW [63] and Proof of Stake (PoS) [64] to secure transactions. This work was supported by using three distinct protocols:

- **BitTorrent** - BitTorrent is used to the file-sharing.
- **Ethereum** - Ethereum is necessary to understand smart contracts and capabilities. At this point, the blockchain comes into the process.
- **TeleHash** - TeleHash is used to make the peer-to-peer messaging because it is decentralized and secure. [65]

As PoC for ADEPT, researchers have deployed these three protocols into a commercial washing machine (Samsung W9000) that was programmed to work with the ADEPT system, making an ”Autonomous Washing Machine Orders Detergent” [66]. The goal is to automate the process of ordering supplies. This process makes use of smart contracts to define the commands to receive a new batch of supplies. This way, the device can order and pay by itself when the detergent capacity is low (blockchain manages the payments). Later, the retailer receives the notice that the detergent has been paid for and ships it. Moreover, the washer owner can also receive notifications of its smartphone's purchase details via its home network.

Another use case consists of a decentralized advertising marketplace using Large Format Displays (LFD) to share and publish content without a centralized controller. The concept consists of an LFD, or, more commonly, a conventional display, where users can share the screen with anybody. Users have to choose the LFD where the advertising will be published and choose the advertisements (video files served by BitTorrent). Then, the advertiser receives the request through peer-to-peer messaging by TeleHash. After this, the content is shared and published. Finally, the advertiser receives the analytics, confirms the approval, and finalizes the payment.

2.1.2 Multiparty Computation

MPC is a sub-field of cryptography, which was formally introduced as a form of secure two-party computing in 1982 and generalized in 1986 by *Andrew Yao's* [67] [68] [69].

The idea behind MPC is to perform computations privately. In this case, suppose that N parties want to compute a function

$$f(t_1,...,t_N) = S$$

where the party i is responsible for input t_i.

The goal is that neither party can obtain more information beyond the pair (t_i, s), which means that no one will know the parties' inputs. Each party knows only the output of the function, which is the answer to the requested problem. For example, imagine an auction where the only bid revealed is the highest. In this situation, it is possible to derive that all other bids were lower than the winning bid. However, this should be the only information revealed about the losing bids. MPC must have correctness - each party receives the correct output, which implies that the party with the highest bid is guaranteed to win, and no party, including the auctioneer, can change this [70].

We can see an image that illustrates MPC in the Figure 2.1.

Most MPC protocols are based on secret sharing or Garbled Circuits (GC) [71]. MPC based on secret sharing refers to methods for distributing a secret to a group of participants, where each participant has a piece (share) of the secret. The only way to reconstruct the secret is with a sufficient number of shares combined: a threshold cryptosystem $(t+1, n)$, where n is the number of parties and $t+1$ is the minimal number of parties to decrypt a secret encrypted with threshold encryption.

Regarding GC, it is a technique to do MPC for two parties. Anecdotally, we will call Alice the generator, which means that she will make the circuit, and we can call Bob the evaluator, which means he will evaluate the circuit. At the beginning of the protocol, both parties agreed on the same circuit. Alice generates a garble table for each of the logic gates in the circuit and sends the garbled

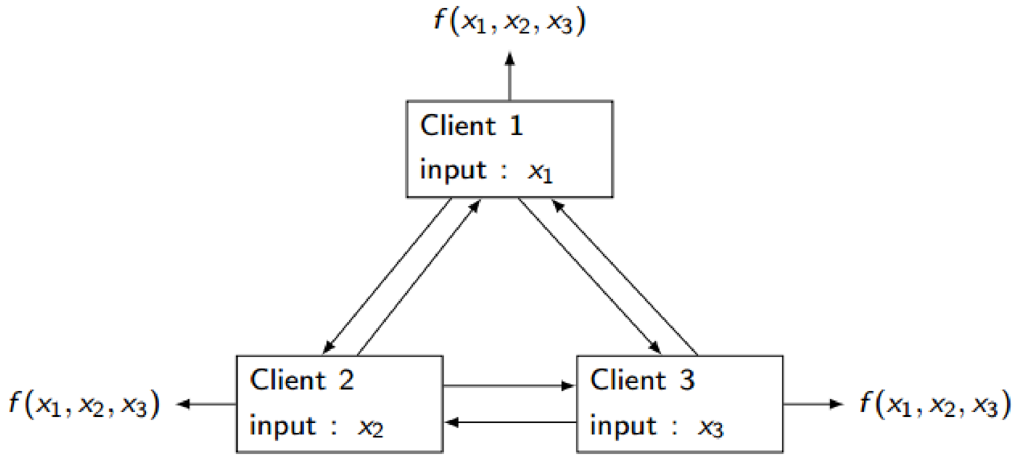

Figure 2.1: MPC without a trusted third-party

circuit to Bob, along with her input values, also encrypted in a compatible way with the encrypted circuit. Bob evaluates the circuit using the garbled circuit protocol decrypting one entry from each of these. Bob uses a technique called ”1-of-2 Oblivious Transfer (OT)” to learn the encrypted form of his inputs without letting Alice know which inputs he obtained. Bob runs the encrypted circuit on the encrypted data, gets the answer, and passes it along to Alice.

2.1.2.1 MPC's applicability

There are many real-world examples for MPC, including the millionaire problem, one of the most well-known real-world application cases. It was first introduced in 1982 by Andrew Yao [67] and consisted of two millionaires who intended to discover the richest among them without revealing their wealth or any other type of additional information to the other party or third parties. The MPC protocol solves this problem without the need for a third party involved.

Assuming that we have three parties: Alice, Bob, and Charlie. Each party uses respective inputs x, y, and z, denoting their salaries. The goal is to find the highest salary of the three, without revealing their respective salaries. Mathematically, this can is achieved by computing:

$$f(x,y,z) = max(x,y,z)$$

Each party will share his input without revealing it to anyone. At the end of the protocol, each participant will get only the function f, without getting anything else about the other party's input, i.e., there are no secret inputs revealed. The security of such protocols concerns the ideal model where f is computed by a trusted party T_f. During the execution of a protocol, the parties cannot get information about the other parties' inputs. A third-party T_f computes a function f receiving the parties' inputs and then computes f, and finally sends the output back to the parties.

Suppose we have Alice and Bob, and both want to know who is richer without revealing to each other or a trusted third party the amount of money they have or any additional information. The function $f(x1,x2) : if\ x1 > x2 = Alice$ else $x1 < x2 = Bob$ computes the inputs and returns the name of the richest (we can see the example in the Figure 2.2. Alice knows that he is richer than Bob but does not know how much money he has, and Bob also knows that Alice is richer but does not know how much money she has. Therefore, in this protocol, the data's privacy is preserved as they never reveal their salary.

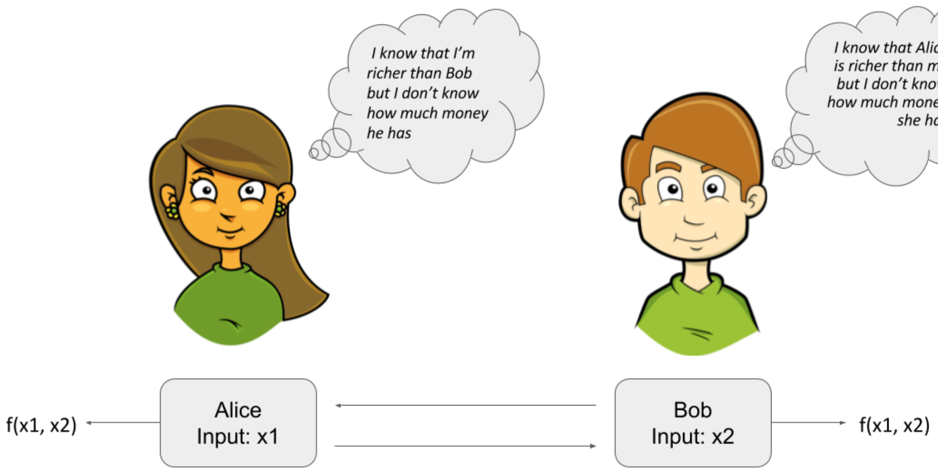
I know that I'm richer than Bob but I don't know how much money he has
I know that Alice is richer than me but I don't know how much money she has
f(x1, x2)
Alice Input: x1
Bob Input: x2
f(x1, x2)

Figure 2.2: MPC - Millionaire's problem

In addition to the millionaire's problem, there are other potential applications of the MPC, such as secret voting, oblivious negotiation, and database queries [67].

Secret voting consists of several m members having globally to decide on a yes-no action. Each member has to choose an option xi and the result is computed by the function $f(x1,x2,x3,...,xm)$. In turn, this function gives the final result without disclosing any other members' opinions and thus preserving privacy.

Regarding oblivious negotiation, anecdotally, Alice tries to sell Bob a house, each with a negotiation strategy. Alice has possible strategies numbered as $A1,A2,...,At$ and the same for Bob as $B1,B2,...,Bu$. The result (no deal or sell at x dollars, for example) is once the basic strategies Ai,Bj used are determined. The result is wrote as $f(i,j)$. This way, it is possible to complete the negotiation obliviously, as Alice will not gain any information on Bob's negotiation tactics, expecting that it is consistent with the outcome, and vice-versa.

The last problem focuses on privately querying a database. Suppose that Alice wants to compute a function $f(i,j)$ and Bob $g(i,j) = constant$. Bob does not know anything about i in the end. If we assume Bob as a database query system, with j being the state associated with the database, Alice can perform a query with the number *i*, and then, she can get an answer without getting any other information besides the data strictly required by her query. Conversely, the database system does not know which element was queried by Alice, allowing users to preserve their privacy while avoiding data leakage from the database system.

Patients may want to access their clinical records, and they can use their secret DNA code to make a query to a medical database of DNA-related diseases. However, patients do not want the hospital, and potential others, to know their DNA and health status. Simultaneously, the hospital does not want to disclose its entire DNA database to the patient because it must preserve privacy. In the same path, social scientists and researchers need to do data analysis, often being sensitive and private leads to legal restrictions and privacy issues. Techniques like Homomorphic Encryption (HE) and MPC can solve this for extensive scientific analysis [72], as both techniques offer private computing.

2.1.2.2 MPC Frameworks Analysis

Several frameworks implement MPC with different protocols, programming languages, number of participants, and security levels. Follows the description of some of the most known MPC libraries:

Sharemind
Sharemind is a framework for privacy-preserving computations that allows to process an input without compromise their privacy. It works as a secure MPC system divided into three parts: input parties, computing parties, and result parties. The input parties apply secret sharing to their data and send a share to each of the three computing parties through the established secure communication channels. Each secret is divided into three shares using additively homomorphic secret sharing techniques and does not reveal any information. After all the data has been sent to the computing parties, the result parties can request these data computations. Each result party can invoke, on the stored data, the execution of algorithms previously inserted in the Sharemind platform. Each computing party runs the algorithms on its shares and sends the results obtained to each result party that requested the computation. Finally, each result party joins the shares it received from each computing party to build the final result.

It consists of a computation runtime and associated programming library to create applications that process personal data, enabling users to develop and test their custom privacy-preserving algorithms. As a result, users can develop secure MPC protocols without the explicit knowledge of all implementation details and allow developers to test and add their protocols to the library, as Sharemind is an open-source project [73]. The experimental Sharemind Software Development Kit (SDK) contains the *SecreC 2* programming language that separates public data and secrets on a system type level and an emulator that developers can use to estimate the running time of their applications in a fully secured environment. *SecreC* programs are fully compatible with the Sharemind Application Server, which provides full cryptographic protection and supports enterprise applications [74].

SPDZ
SPDZ implements a general MPC protocol, secure against an active adversary corrupting up to n-1 of the n players [75].

The processing model implemented by SPDZ is as follows: a) an offline phase that generates pre-computed values that can be used during the online phase, and; b) an online phase that performs the secure computation, executing the designated functions.

SPDZ-2 [76] library was introduced in 2016, and the framework includes some examples available for users to test, such as the millionaires problem. In 2020, the authors claimed that this software is not under active development anymore and that there are two successor projects, namely SCALE-MAMBA that aims to provide an integrated framework for computation in prime order fields with both dishonest and honest majority and MP-SPDZ [77, 78] that aims to provide benchmarking of various protocols, while preserving all the functionality of SPDZ-2.

FairplayMP
Fairplay [79] is a full-fledged system that implements generic Secure Function Evaluation (SFE). SFE allows two parties to compute data without any trusted party jointly. However, the Fairplay system uses Yao's GC and only supports secure communication between two parties. FairplayMP is an extension that appears to counter this limitation and introduce multi parties because cryptographic protocols for the multiparty scenario are entirely different than protocols for the two-party case [80]. It implements secure computation using Yao's circuits and secret sharing techniques.

ABY
ABY [81] combines secure computation schemes providing three different secret sharing schemes: Arithmetic sharing, Boolean sharing, and Yao's GC, allowing two-party computation to be secure. It allows the pre-computation of almost all cryptographic operations, and the authors claim that it has highly efficient conversions between secure computation schemes based on pre-computed OT extensions.

This library only considers semi-honest (passive) adversaries. It assumes a computationally-bounded adversary who tries to learn additional information from the messages seen during the protocol execution. The adversary cannot deviate from the protocol, so the authors consider that ABY is secure against passive insider attacks by administrators or government agencies or when the parties can be trusted not actively to misbehave.

The implementation of ABY [82] includes some sample tests for users, such as the millionaire's problem, secure-computation Advanced Encryption Standard (AES) and euclidean distance.

TinyLEGO & DUPLO

TinyLEGO [83, 84] focuses on general secure two-party computation based on GC. The authors claim that it provides privacy-preserving even if one of the two parties acted maliciously during garbling. TinyLEGO belongs to the LEGO family but has optimizations, such as getting different outputs to both parties with minimal overhead, using an XOR-homomorphic commitment scheme, an authentic, private, and oblivious garbling scheme, and a 2-correlation-robust and collision-resistant hash function.

DUPLO [85] is based on TinyLEGO but with more stable code and faster. It supports the same circuit format as TinyLEGO, as well as an enhanced version for more flexible computation [86].

2.1.2.3 Comparison between MPC frameworks

Framework	Programming Language	Techniques	Security	Number of participants	Year of creation
Sharemind [73, 74]	SecreC (C++)	Secret Sharing	Semi-Honest	3	2006
SPDZ-2 [87, 88]	Java, C++, Python	Secret Sharing	Malicious adversaries	$\geqslant 2$	2016
MP-SPDZ [77, 78]	C++/Python	GC or Secret Sharing	Malicious, Semi-Honest, with Dishonest or honest majority	$\geqslant 2$	2020
FairPlayMP [80]	SFDL (Java)	GC and Secret Sharing	Semi-Honest	$\geqslant 3$	2006
ABY [81, 82]	C++	Arithmetic sharing, Boolean sharing, and Yao's GC	Semi-Honest	2	2014
DUPLO [85]	C++	GC	Malicious adversaries	2	2016

Table 2.1: Comparison between MPC frameworks

We decide to overview different frameworks with different languages and protocols. Table 2.1 represents a comparison between MPC frameworks, and we can see that FairplayMP, DUPLO, and MP-SPDZ use GC, whereas Sharemind and SPDZ use additive secret sharing over a ring. Finally, FairplayMP uses Shamir's secret sharing * [89, 90]. Lastly, ABY uses a combination of Arithmetic sharing, Boolean sharing, and Yao's GC.

The main application for GC seems to be secure two-party computation. For more than two parties, frameworks are using secret-sharing schemes [91]. All these frameworks support a similar set of primitives, including multiplication, comparisons, and equality testing. Programming on these platforms either uses a specialized language or a standard programming language and library calls, depending on the platform [90].

Regarding the security levels, *João S. Resende et al.* [92] evaluate some of these frameworks, namely ABY, DUPLO, SPDZ-2 and TinyLEGO (that is one previous version of DUPLO). The authors conclude that ABY, DUPLO, and TinyLEGO are secure for some input size (from 0 to 9). For this reason, we will use ABY on the implementations that use MPC in this thesis because it is considered secure and still has maintenance at the time of this thesis.

2.1.2.4 Future Research Directions on MPC

This section discusses the main research questions and summarizes the identified challenges in the MPC area.

Most of these frameworks are stated as academic testing prototypes, not ready for deployment in real-world scenarios. Many authors state that the code is not fully secure because security-related issues are not fully implemented and do not have security reviews. There is still a need to combine the best (safe) MPC solutions and optimize them for production and real use cases. Moreover, we can easily launch a MiTM attack on the libraries as there is no authentication. For now, establishing authentication using these libraries requires using something like Virtual Private Network (VPN) connections or authenticated channels between the hosts, then the traffic is encrypted and authenticated.

A new research path consists of improving the MPC implementations to enable novel scientific data methods by creating new tools that will make these techniques accessible to social scientists. This pathway points to the need for a closer examination of automatic data-matching between separate datasets with a private set intersection, improving fixed-point integer conversion for decimal data values used in computation and other privacy-preserving applications. To summarize, the ultimate goal is to achieve this without disclosing private information, i.e., each party's inputs. Along with this, the number of devices and data in IoT is growing, so it would be essential to have a filter to identify non-sensitive data, making tools that detect sensitive versus non-sensitive data. Research work already exists in this direction, namely *SecureML* [93], which uses MPC and focuses on

*Shamir's Secret Sharing is a form of secret sharing, where a secret is divided into parts, giving each participant a random part of the secret, where some of the parts or all of them are needed to reconstruct the secret. Sometimes, it uses a threshold scheme to define *k* parts that are sufficient to reconstruct the original secret, as it can be impractical to have all participants together. [89]

privacy-preserving machine learning for linear regression, logistic regression, and neural network training using the stochastic gradient descent method.

Marcin Andrychowicz et al. [94] propose an exciting research direction for *MPC on Bitcoin* where Alice and Bob can determine who is the wealthiest one based on who has more coins. However, this is only possible if each party is interested in proving that it is the wealthiest one because every participant can easily pretend to be weaker than it is and ”hide” its real wealth by transferring it to another address under its control. This method helps to obtain fairness in specific multiparty protocols. For example, Bitcoin uses an attractive way to build ”timed commitments”, where the committer has to reveal his secret within a specific time frame or pay a fine. In our opinion, this can be a possible research direction not only in the millionaire’s problem but in other problems that are isomorphic, just with varying underlying contexts. Also, there are other open problems in the MPC, such as constructing protocols that are secure against ”malleability” and ”eavesdropping” attacks.

2.1.3 Multiparty Computation and Blockchain

With the increasing spread of IoT, there is a need to create decentralized and private platforms. Combining MPC with blockchain technology can be a significant advance in this area, as it may be possible to create platforms that allow privacy preservation (data remains encrypted, even in use) and resilience. However, some questions remain unanswered: if it is possible to design a decentralized platform without relying entirely on a trusted third party or building a fully decentralized protocol to sell secret information without allowing sellers and buyers to cheat.

2.1.3.1 Enigma

Enigma [21] combines the use of MPC and blockchain technologies. This section describes the Enigma technology and the associated use cases with real-world applications.

Enigma is a peer-to-peer network that enables different parties to store and run computations on data while keeping the data completely private. This model works in parallel with an external blockchain technology. Similar to Bitcoin, Enigma removes the need for a trusted third-party.

Enigma’s primary motivation focuses on avoiding centralized architectures that might lead to catastrophic data leakage that would result in the loss of privacy. It connects to an existing blockchain and off-load private and intensive computations to an off-chain network. The blockchain (public tasks) and Enigma (private and computationally intensive tasks) execute the code.

Opposing blockchain, which only ensures correctness, Enigma’s execution provides privacy and correctness simultaneously. One of its main features is its privacy-enforcing computation, as Enigma can execute code without data leakage while still ensuring correctness. As heavy-duty computations are a known issue for blockchains, Enigma only allows the broadcast of running computations throughout the blockchain to avoid heavy processing.

Although blockchains are not general-purpose databases, they can be used to store information strategically. Enigma has a hash table distributed outside the decentralized chain that uses blockchains to store data references (not for real data). However, the client-side must encrypt private data before programming the blockchain's storage and access-control protocol, which will act as public proof of the authorization scheme.

Enigma Application Cases

There are multiple fields of application for MPC, where privacy-preserving is a concern. In this section, we describe some of the most relevant domains we envision can be applied.

Applicability in IoT seems straightforward, as we can store, manage, and use highly sensitive data collected by IoT devices in a decentralized, trust-less cloud. The Crypto Bank is also a field where the intimate details have to be anonymous, so we can run a full-service crypto bank without exposing its internal design and implementation. The blockchain's autonomous control allows users to take loans, deposit cryptocurrencies, or buy investment products without publicly revealing their financial situation.

In line with the millionaire's problem, where n-parties want to know if they are wealthier than the others, without exposing their financial status to each one, there is blind e-voting. The latter case maintains each voter's privacy, and the actual vote count can potentially remain private.

Another Enigma application is the n-factor authentication, where voice, face, and fingerprint recognition are all stored and computed on Enigma. As private contracts support the access-control, only the user can access its data.

Furthermore, private contracts allow us to share some data with a third-party securely. It is possible to define some contracts restricting access to data, maintaining and enforcing control and ownership. The shared data on MPC is always reversible, as third parties do not have access to actual raw data and can only run secure computation. Private contracts also support identity management. When users want to log in, the system executes an authenticating private contract to validate the users and link to their real identity with a public pseudo-identity, making this process wholly trust-less and privacy-preserving. This way, the authentication and identity storage are fully anonymous, and users on Enigma only have to secret-share their personal information required for authentication.

For data protection, privacy-preserving approaches should be paramount to companies, as they hold large volumes of potentially sensitive user data that is a potential target for criminals. With Enigma, companies can use data to provide personalized services and match individual preferences without storing or processing the data on their servers, thus removing the security and privacy risks. By doing so, Enigma can protect companies against corporate espionage and rogue employees. Note that employees can still use and analyze data for the user's benefit while enforcing agreed consents. With these solutions, companies can potentially provide access to the data while preserving security and privacy.

A data marketplace can be a potentially compelling case because, for example, a pharmaceutical company looking for patients for clinical trials can search genomics databases for candidates. In this process, consumers can sell access to their data with guaranteed privacy, autonomous control, and greater security. The marketplace would eliminate tremendous friction between companies and individuals, reduce customer acquisition costs, and offer a new income stream.

2.1.3.2 Future Research Directions

Computation on encrypted data has been slow in practice, and it remains an active research path [95].

Additionally, Enigma entails processing transactions without knowing their contents that might provide an alternative way to achieve similar accountability benefits while supporting transactions. However, this approach does not preclude the possibility of a validator favoring transactions based on a bias because it can identify the transactions with help from colluding peers, even if the transactions are encrypted [96].

Enigma's combine three paradigms: secret-sharing, MPC, and Peer-to-Peer (P2P), which opens new possibilities to address current open issues on data privacy and the growing liabilities faced by organizations that store or work on large amounts of personal data.

2.1.4 Summary

This chapter has provided a summary of the multiple concepts of secure computation in different approaches. The first sections (2.1.1 and 2.1.2) presented some concepts of secure computation such as secure MPC that consists of exchange data anonymously without a trusted third-party, or blockchain that is a secure way of online transaction. We described some concepts and their applications and combined both (if any) and with the IoT. In the secure MPC section, we overview some MPC frameworks and analyze each functionality.

We discovered some interesting applications which show that there are some attempts developments based on the combination of some of these concepts. The main finding applications were:

- Blockchain for IoT: IOTA and ADEPT (described in the section 2.1.1.3);
- Blockchain with secure MPC: Enigma MIT's (described in the section 2.1.3.1).

As we describe in Section 2.1.1.2, there are several limitations to integrate the blockchain technology with IoT. Section 2.1.1.3 describes a solution that consists of using fog computing as a way of decreasing latency, having ”local awareness” and mobility (including vehicular mobility). As cloud computing limitations are undesirable for IoT, it is essential to do the integration between IoT and blockchain using the fog computing with IoT.

However, there are unsolved problems and open issues that we described as future research directions in the sections 2.1.2.4 and 2.1.3.2.

2.2 IoT Identity Management

There is no direct integration between traditional human-centered IdM systems and IoT environments [97]. Humans are distinguished by their physical characteristics, nationality, and personality, to name a few, and have an identification document to prove their identity. The human identity can be composed by the combination of the identification document, which has personal information (name, nationality, or birth date) and other characteristics that define the identity of a person, such as preferences (food, clothes, books) or reputation among the community (honest or reliable) [98]. Digital identity is equivalent to a human's physical identity when used for identification and transactions. However, some services and applications do not require all the information associated with the users. For example, e-bay only needs to know if the seller's reputation is good, and the seller only needs to prove that it controls his digital identity. In this case, attributes like date of birth, gender, or nationality are not required (less important). Once the link establishment between the private entities and their online interactions, digital identity involves authentication. Subsequently, human authentication methods can be classified into three categories [99]: *something you have*, *something you know* and *something you are*.

However, for devices, it is required to define a digital identity. Therefore, device identification, authentication mechanisms, and relevant forms of authorization are required to address privacy and security issues in IoT. With a unique and strong identity, sensors and devices can be authenticated online, promoting safer communication between devices, services, and users, thus proving their integrity.

Theft, tampering, and disguise are some of the issues that challenge IoT identity protection. Also, there is a lack of clear definitions of how sensors or devices are identified, represented, searched, and accessed in IoT. This gap makes IoT vulnerable to multiple identity attacks, such as a Sybil attack, when a sensor or device illegally uses multiple identities. Thus, there is a need to review the different traditional security solutions to determine their feasibility and applicability in IoT.

2.2.1 Traditional IdM

IdM is a set of processes and technologies used to protect the identity of an entity (user or device), ensure the quality of identity information (identifiers, credentials, and attributes), and provide authentication and access privileges to information systems within limits defined by an organization. An entity can have multiple identities, and each identity can have different attributes; each attribute can be unique or non-unique. We call this section *Traditional IdM* as it is not focused on IoT but on generally solutions for IdM.

Multiple services integrate the IdM concept, such as: Active Directory Management; Service Providers; Identity Providers; Web Services; Access Control; Digital Identities; Password Management; Single-Sign On (SSO); Security Tokens Security (STS); OpenID; WS-Security; WS-Trust; Security Assertion Markup Language (SAML) 2.0; OAuth and Role-Based Access Control (RBAC).

An IdM system consists of a three-tier architecture: User or Device, Identity Provider (IdP) and Service Provider (SP) [98, 100, 101].

One of the most common ways to manage authentication and authorization of individuals and devices is by using *Federated IdM* [102].

SAML and *OAuth* represent the two most widespread approaches.

SAML is an XML-based standard developed by the OASIS Security Services Technical Committee to exchange authentication and authorization data between security domains [103]. *SAML* specifies the issuance of a token signed by an IdP, where the SP needs to maintain the public key of the issuer to check the validity of the signed token and its timestamp. However, users that own the signed token can impersonate the real owner, so this token should have a short expiration time.

OAuth 2.0 and *OpenID Connect 1.0* are two standardized authentication and authorization frameworks used by most services. The *OAuth 2.0* framework delegates conditional authorization. That is, resource owners authorize temporary access to a predetermined set of resources without revealing their credentials. Access tokens are provided to third-party clients by an authorization server with the resource owner's approval. Then, clients use their access tokens to access protected resources hosted on the resource server. Users commonly use this service to access third-party websites using their Facebook, Google, and Twitter accounts, without exposing their credentials [104].

OpenID Connect adds an identity layer to the *OAuth 2.0* [105]. This implementation of *OAuth* is based on the use of bearer tokens, which have a limited validity in time and are verified by the signature made by the IdP.

These solutions do not provide security and privacy entirely to the federated online scenarios because the IdP acts as a SPOF, as it can impersonate the users because it is involved in all authentication processes and can track its users.

Some research works focus on solving identity management problems. *Caroline Chibelushi et al.* [106] propose a healthcare IdM framework built into Mobile Ad-hoc Network (MANET), assuming the devices are connected wirelessly and allowing users and devices to be distinguished based on personal identifiers and device profiles, respectively. The structure also considers bandwidth limitations, ensuring the exchange of the minimum amount of information. A sandboxing [*] technique is employed to protect user content when sharing a device. Most IdM systems focus on PKI's, which links public keys to entities' identity (like people and organizations). As DigiCert claimed, a PKI

[*]Sandbox can be an essential layer to the protection system. A system can concentrate its operations in a restricted area where all unreliable programs, records, and activities can run completely isolated from the computer's operating system. [107]

security solution, when properly implemented, provides strong device identity and encryption for in-transit data and protects devices and networks from exploits.

Nowadays, the *OAuth* and *SAML* for example, are often the basis for building an IdM system, such as *Shibboleth* or *Keycloak*.

Shibboleth [108] is an open-source project that provides federated SSO and attribute exchange systems implementing widely used federated identity standards (as described in the Section 2.2.1) by exploiting *SAML*. As far as we know, this is the second most cited and used system, preceded only by *OpenID* [109]. To use a SP connected to a federation, users must authenticate using their organizational credentials (Shibboleth Identity). This way, an organization (IdP) will pass the least amount of information to the SP manager, whether allowing user access. As a result, authenticated users will have access to any SP connected to this federation. This tool focuses on user authentication based on attribute exchange [110]. *Shibboleth* consists of several individual components: an IdP, SP, and Discovery Service. Users may choose to deploy one or more of these components depending on their needs. Currently, *Shibboleth* is not being considered as an IdM for IoT scenarios because, although SSO may be useful as users only need to authenticate once to interact with various devices, the traditional Web 2.0 SSO does not meet certain IoT requirements [111].

Keycloak [112] is an open-source Identity and Access Management (IAM) solution focused on modern applications and services. It supports User Federation, Identity Brokering, SSO, and Social Login. SSO allows a user to log in only once and then access all systems configured in *Keycloak*. *Keycloak* is based on standard protocols and provides support for *OpenID Connect*, *OAuth 2.0*, and SAML.

PRIME [113] is a system that focuses on effectively managing and protecting users' private data. This system uses *Idemix* [114], which allows the creation of anonymous credentials that can be deactivated.

Even with the existence of SSO, such as *OpenID* or *Shibboleth*, security and privacy still need to be fully addressed in IoT [115]. These problems lead to the need to analyze IdM systems for IoT and define an identity for *things*.

2.2.2 Identity of Things

There is an area of endeavor in IoT that includes assigning a Unique IDentifier (UID) with metadata associated with devices and objects (items) that allows them to connect and communicate with other entities on the Internet effectively. Unlike the three categories of human multi-factor authentication (*Something you have*, *Something you know* and *Something you are*), the approach in IoT is more complex. [116] present an idea for Identity of Things based on four categories: *inheritance*, *association*, *knowledge* and *context*.

Inheritance category is equivalent to biometrics in humans, so it is necessary to find an identical mechanism that identifies devices. The suggested mechanism is the Physically Unclonable Function (PUF) [117–119]. However, this mechanism has some known attacks, such as those described in previous works [120–123]. It is also not as flexible as the other categories, as it depends on the chip/hardware manufacturers.

The *association* and *knowledge* categories do not have as many hardware requirements as *inheritance*. *Association* is equivalent to the *"something you have"* category of human multi-factor authentication, but it is not easy for an IoT device to process something external (such as a hardware token). Similar to the *"something you know"* category, the authors present the *knowledge* category where there is, for example, the phone's International Mobile Equipment Identity (IMEI).

However, what attracts more attention in IoT is the *"context"* category, which the authors also refer to as the fourth category of authentication methods. *Context* refers to the device's environment, such as real-time device location.

2.2.3 Ongoing research on privacy-preserving IdM for IoT

The power and bandwidth limitations of IoT devices make the integration of IdM and IoT much more complex and challenging [116].

There are many IdM systems for IoT, and we have reviewed some of the most cited projects. Researchers have been working on IdM solutions but are not entirely focused on solving privacy issues. Although there are new frameworks [110] that guarantee specific defined security goals, more implementation and evaluation details are lacking for analyzing the privacy properties it contains. Another approach focuses on the user-centric IdM framework consisting of user identity, device identity, and the relationship between them [124]. It correlates a user's identity with a device's identity but does not address privacy issues.

In this thesis, our focus is primarily on properties that can help preserve data privacy, especially confidential data, by addressing the data minimization principle, a central aspect of the recent GDPR [125].

Partial identity is a fundamental principle of privacy adopted by previous works [126, 127]. However, these works do not preserve anonymity because a fully disclosed certificate identifies entities to other parties. For example, most works use X.509 certificates as credentials to real model identities and *SAML* security tokens to encode partial identities to prove ownership of specific attributes [128].

As privacy is a focus of our research, we have decided to highlight systems and projects based on the Anonymous Credential concept [129], where we can display credentials without compromising privacy. The idea behind the concept is that users can obtain credentials and display some of their properties without revealing additional information or tracking. Each token, as well as credentials, is designed to be used only once. There are several examples, such as subway tokens, electronic money (e-cash), movie tickets, and access passes for online services. Repeated use of the token reveals the user's identity, but several tokens used by the same users are unlinkable.

For example, in a movie theater, there are age restrictions on buying tickets for certain movies. If the clients want to prove that their age is over 18 years old, the movie theater does not need to access more information, such as an identification card with all personal information, including a full birth date. Instead, it only needs to obtain a proof that the clients' age is over 18.

There are two implementations that address privacy issues through the concept of anonymous credentials: *Idemix* [114] and *UProve* [130]. Through protocols and cryptographic mechanisms, the schemes implemented by each model allow the presentation of credential authentication through credentials and proofs of attributes, preserving anonymity.

Recent works [128, 131] use *Idemix* in their implementations. There is a holistic IdM system [128] that handles different IoT scenarios that require traditional online access control and authentication, along with a claims-based approach to privacy-preserving P2P interactions. The system follows a claims-based approach with attribute-based credentials. This project has been tested and implemented within the European research project *SocIoTal*. The IdM system features the IBM's cryptographic library *Idemix* [114], providing a privacy-preserving solution that addresses IoT scenarios in which consumers and vendors should not be only traditional computers, but also smart objects (e.g., smartphones). In addition, *SocIoTal IdM* has been integrated with *FIWARE Keyrock IdM* [132] to support traditional IdM management operations in scenarios where claims-based access is not required. The implementation is in Java, and, as described in the official implementation of this project [133], five main components make up the IdM. However, the *SocIoTal* [128] system is not suitable for resource-limited devices because, despite being a real redeployment to integrate *Idemix* into IoT, it is intended for Android devices and uses Java programming language, which requires high computational and memory resources to run.

On the other hand, *Jose Luis Canovas et al.* [131] proposes a solution for authentication and authorization with privacy-preserving that uses the concept of anonymous credentials. This concept aims to have the IdM inside a device, avoiding consulting external trusted third parties, and even check attributes; it has zero-knowledge proofs, and, for example, vehicles could authenticate their owner by verifying a proof from their wearable, such as a smartwatch. The authors give an example that focuses on the verification of whether someone inquiring is entitled to get some information, e.g., an accredited person, such as a policeman, can issue information about the vehicle's, and in that case, the vehicle would disclose its owner's relevant information, such as proof of passing technical inspections. Otherwise, if the device requiring information is not an accredited person, some sensitive information is restricted. In brief, a device may or may not restrict the sharing of certain information (minimum disclosure).

2.2.4 Blockchain

The Section 2.1.1 describes the blockchain concept and the limitations of integration with IoT. This section presents the motivations of using blockchain with IoT, and an overview of blockchain-based

IdM systems.

2.2.4.1 Motivation for using Blockchain for IdM in IoT

Most previous works related to IdM, authentication, and authorization focus on heavy and complex communication protocols in terms of computation and memory requirements and have SPOF problems that can compromise the entire system. PKI-based solutions are complex, expensive, and not easy to manage, which conflicts with IoT integration due to resource constraint issues on devices. PKI also relies on a Certification Authority (CA) that represents a SPOF problem. If an attacker could create a fake CA, it would provide digital certificates that would be accepted as true by many browsers. A browser can then claim that a site is legitimate when it is a fraud. By using fake CAs and exploiting the MD5 algorithm flaw, hackers can use the well-known Domain Name System (DNS) flaw to create unidentifiable phishing attacks. It is essential to eliminate the SPOF problems inherent in trusted third parties and create a decentralized solution. Adopting a standardized peer-to-peer communication model for processing hundreds of billions of device-to-device transactions will significantly reduce the costs of installing and maintaining large centralized data centers. It will also distribute computing and storage needs among billions of devices that form IoT networks, preventing the SPOF problems [134].

Blockchain enables creating more secure network meshes in which IoT devices are securely interconnected, avoiding threats like spoofing and device impersonation. As blockchain registers each legitimate node, devices will be able to identify and authenticate themselves. This block-based approach is more agile than other approaches, and each registered identity can be associated with the device's public key, allowing for a more secure communication scenario. However, it is possible to suffer a Sybil attack with blockchain, so it is essential to know that the nodes are secure. When there is no central authority, a reputation system is required. User accounts created on blockchain-based reputation systems do not have their real identity revealed. Attackers may leave the system after attempting to inject fraudulent subjective information but will not rejoin and create a new account to launder their previous rating history [135]. Leveraging blockchain's capabilities, each device with an identity has an immutable reputation and history when its certification agency audits the device and registers its identity with blockchain from birth. Otherwise, centralized systems have a hierarchical context addressed (device @ host, with the host gaining its identity by assigning an IP address or registering a DNS) [136]. As already described, blockchain also provides capabilities such as traceability, reliability, autonomic interactions, and interoperability between devices [137].

2.2.4.2 Blockchain-based IdM systems

In this section, we systematically review blockchain-based IdM systems. Past works critically analyze and compare the different blockchain-based IdM and authentication systems from 2014 to 2018 [138].

These solutions can be categorized into *permissioned* and *permissionless* blockchain [139]. The authors highlight two systems in each of the categories: *uPort* [140] and *Sovrin* [141].

Another categorization [142] for IdM solutions can be done between: *Self-Sovereign identity* and *Decentralized Trusted Identity*. The authors highlight three Distributed Ledger Technology (DLT)-based IdM schemes, namely, *ShoCard*, *uPort* and *Sovrin*, and evaluates their benefits and shortcomings by providing a detailed description of the three systems.

While *ShoCard* focuses on digital identity for humans, *UniquID* [143] is similar but attempts to fill the gap between human and digital entities [144]. For this reason, we also include this system in this description.

A recent paper proposes a decentralized privacy-preserving healthcare blockchain in IoT [43]. It is of utmost importance to describe this system because it also focuses on the privacy-preserving properties and healthcare use cases.

Therefore, a detailed description of five blockchain-based IdM systems will be provided in this section: *Sovrin*, *uPort*, *ShoCard*, *UniquID*, *A Decentralized Privacy-Preserving Healthcare Blockchain for IoT*.

Sovrin

Sovrin [141] is a public permissioned distributed ledger dedicated to self-sovereign identity.

The consensus protocol used in *Sovrin* is called *Plenum* and is an improvement over Redundant BFT [145] based on the BFT protocol.

The choice for a permissioned blockchain and applying different consensus protocols are possible enhancements for low-cost computation, thus reducing the energy cost of running a node and improving transaction throughput. All of these choices are beneficial for the integration between *Sovrin* with IoT, specifically for resource-limited IoT devices.

In *Sovrin*, a user can generate any number of required identities. This way, it can guarantee privacy because identities are separate, unlinkable, and controlled by a different asymmetric key pair. These identities follow the Decentralized Identities (DID) specification, a set of features that uniquely define objects. There is no central register to give to an entity a ”positive signal” in the data's validity. DID is entirely under the authority of the user.

Sovrin integrates unforgeability, performance, unlinkability, and a distributed ledger, adopting the best practices of both *Ethereum* [146] and BFT protocols. However, one of blockchain's problems is that it is easy to violate privacy in a public ledger by correlating transactions to make inferences about users. To mitigate this risk, the system includes the concept of anonymous credentials [147], which gives users full control over all aspects of their identity.

uPort

Unlike *Sovrin*, *uPort* [140] does not provide a full stack for managing distributed ledger identities for devices. It is based on an Ethereum smart contract to design a digital identity model and is a public permissionless blockchain where anyone can be a validator node. *uPort* allows users to register their own identity on Ethereum, submit and request credentials, sign transactions, and manage keys and data securely. The system aims to abstract end-user public key cryptography to make the user experience intuitive. It provides self-sovereign identity, meaning that users can store their identity data on their own devices and efficiently provide it to those who need to validate it without relying on identity data's central repository. A mobile app holds the user's private key, and a smart contract address acts as its identifier [148]. It ensures identity reliability and usability through a set of operations (key and identity recovery) [139].

ShoCard

ShoCard [149] is a different approach, more focused on user identity and helping individuals and businesses quickly validate identities without using passports or other physical identification documents. It allows it to work as a mobile identification that can be verified in real-time while using a combination of encryption and Bitcoin ledger immutability. Perhaps most importantly, the company claims that identifying information can be verifiable without requiring users to relinquish their data.

Users create their identities on the blockchain using their details, and a known and trusted organization with the ability to verify identities must validate them. Users can use their own identity to travel, and their travel agency can search blockchain and verify that a trusted organization has validated their identity.

In terms of privacy, the authors claim that user data is not stored in blockchain but has its cryptographic proofs to show that it is correct. It also provides selective disclosure to create a key pair for each of the fields that the user is storing in *ShoCard*, so that the user has a private master key and private keys for individual data fields. A private key protects the fingerprint of data on blockchain; therefore, only the user who owns the private key can modify it.

UniquID

Another work called *Uniquid* [143] is building a technology that identifies devices themselves while offline through the use of blockchain technology and smart contracts.

Instead of using PKI, *UniquID* applies PGP and Web of Trust [150] principles. The identification of each device is generated using pseudo-random functions, following the same principle as Pretty Good Privacy (PGP). However, as there are no third parties in this process, *UniquID* devices must rely on a *secure element* [151] to generate their own identity and maintain integrity. Finally, *UniquID*, following the Web of Trust principles, requires a decentralized mutual recognition process using cryptographic signatures. *UniquID* devices must first complete an Imprinting Ceremony [152] with other previously enabled devices, closer to the time of manufacture, to exchange public keys in a secure environment and reduce the risk associated with MiTM attacks.

This system focuses on IoT, and the authentication process is performed between devices without the need for third-party intermediaries, allowing devices to be independent [138].

A Decentralized Privacy-Preserving Healthcare Blockchain for IoT

Ashutosh Dhar Dwivedi et al. [43] feature a decentralized privacy-preserving healthcare blockchain system for IoT. By eliminating PoW, the authors made some adaptations to blockchain to make it suitable for IoT. The authors implement some techniques to preserve privacy in identity, such as a ring signature scheme to anonymize user data. There is still no evaluation or implementation of this system to analyze the system's functionality and if the blockchain adaptation worked, i.e., if it is lightweight enough for IoT.

2.3 Authentication and Secure Communications

Many applications provide identity, authentication, and authorization across multiple contexts. Authentication schemes have been studied in the literature [153–162] and different authors present many taxonomies. *Shubham Agrawal et al.* [163] claim that there are different authentication schemes, named: OTP, Zero-Knowledge proof, Mutual Authentication, Public Key Cryptography, and Digital Signature. On the other hand, *VL Shivraj et al.* [164] refer to a different categorization: mutual authentication schemes, two-party authentication through a trusted party with key exchange, session key-based authentication, group authentication, directed path-based authentication scheme, OTP and SecureID authentication schemes. *Nidal Aboudagga et al.* [165] has presented a document with a taxonomy and research issues in the authentication protocols for ad-hoc networks.

PKI provides important core authentication technologies for IoT. PKI creates digital certificates that map public keys to entities that securely store these certificates in a central repository and revokes them if needed. Any device can verify the integrity and ownership of a public key in this type of infrastructure. Previous studies [166] show that 42% of devices will continue to use digital certificates for authentication and identification in the next two years. The *Secure Sockets Layer (SSL)/Transport Layer Security (TLS)* [167] is an example of an authentication system based on a PKI.

This chapter will cover the authentication methods related to the solutions that we developed throughout this thesis. Therefore, we will separate the authentication types into three different sections: main agreement protocols, one-time password schemes, and proximity-based solutions. We analyze the most relevant solutions in these groups, highlighting those that most interest us for our research.

2.3.1 Key agreement protocols

The ZRTP is a key agreement protocol used on Voice Over Internet Protocol (VoIP). This protocol does not use digital certificates; instead, it uses Diffie-Hellman (DH) keys (also called shared secret keys). The DH method is a specific encryption algorithm for key exchange based on discrete logarithms. At the end of this key agreement, it is possible to generate, through shared secrets, a master key to create a Secure Real-time Transport Protocol (SRTP) Cryptographic Context and so, establish an SRTP stream [168][169]. The DH algorithm alone does not protect against MiTM attacks. During key exchange, to authenticate both peers, the protocol uses a SAS generated during the key negotiation from the DH shared secrets. In the case of a MiTM attack, the devices will end up with different shared secrets and, thus, different SAS. If the SAS is the same, the communication is secure. However, the SAS needs to be confirmed on both users' devices. When the communication is validated, the shared secrets are used to produce new secure sessions in the future, decreasing the computational effort and skipping the user intervention to compare the SAS on the subsequent communications between the same peers [169].

The Password-Authenticated Key Agreement (PAKE) is another method for key agreement but based on pre-shared passwords. It consists of two or more parties establish cryptographic keys based on one or more party's knowledge of a password. *Jean Lancrenon et al.* [170] discusses the three main state-of-the-art PAKE protocols. Password Authenticated Key Exchange by Juggling (J-PAKE) [171, 172] is the most recent PAKE protocol that uses an elliptic curve DH for key agreement and a Schnorr Non-Interactive Zero-Knowledge (NIZK) signatures [*] proof mechanism to authenticate two peers and establish a shared secret between them based on a pre-shared password. Some services still use J-PAKE, such as the Pale Moon Web-Browser, the lightweight Application Programming Interface (API) in Bouncycastle (1.48 and onwards), and Thread (IoT wireless network protocol). There are several known J-PAKE issues, already published by *Mohsen Toorani* [174]. J-PAKE is vulnerable to a password compromise impersonation attack and has other shortcomings concerning replay and Unknown Key-Share attacks. According to the same authors, OpenSSL and OpenSSH integrated J-PAKE, but there were some problems reported during implementation [175], and they no longer use it.

Device Pairing Using Short Authentication Strings [176] is a two-device pairing mechanism based on the agreement and checking of a secret's authenticity using a SAS. This protocol consists of three phases: discovery, agreement, and authentication. When the pairing service starts, the server starts publishing the chosen instance name. The client will discover the name and the corresponding connection parameters [176]. The client and server then use a TLS session to agree on a shared secret using a cryptographic protocol producing a SAS. After this, an authentication phase is used to validate the pairing through a SAS. In this phase, users have to make the SAS comparison through a manual verification (verify that both devices display the same string). If, instead, the server and client support Quick Response (QR) codes, the server displays a QR code with the SAS's encoding, and the client is capable of scanning the value of the SAS and comparing it to the locally computed value.

[*]The Schnorr NIZK proof allows one to prove the knowledge of a discrete logarithm without leaking any information about its value. It can serve as a useful building block for many cryptographic protocols to ensure that participants follow the protocol specification honestly. [173]

2.3.2 Proximity-Based Approaches

Additionally, there is a vast literature related to Location-Limited Channel (LLC). *Dirk Balfanz et al.* [177] presents new schemes for peer-to-peer authentication in ad-hoc wireless networks based on LLC. The authors also describe how to use demonstrative identification to perform pre-authentication over LLC.

Amir Spahić [178] presents an authentication mechanism (pre-authentication phase) which uses context information through LLC using Infrared. *Serge Vaudenay* [179] presents a concept that authenticates a short string, the SAS, through an extra insecure channel. This concept is similar to our proposal; however, it uses a narrow-band authentication channel.

Another proposal [180] is based on a new LLC using biometrics. The protocol efficiently calculates a shared secret key from biometric data using quantization and crypt-analysis. The authors use grip pattern-based biometrics as an LLC channel to achieve pre-authentication in a protocol that sets up a secure channel between two handheld devices.

N Asokan and Philip Ginzboorg [181] presented a key agreement protocol in ad-hoc networks based on PAKE and a location-based key agreement to authenticate through a location-limited channel, such as Bluetooth. However, such a protocol does not seem focused on IoT. It does not have the key continuity feature that is a proper application in this context, as key continuity allows devices to move away after the first pairing and continuously have secure communications on the following connections.

2.3.3 One-Time Password

There are several end-to-end solutions more related to OTP solutions. Some of them, are based on temporary passwords and/or unique numbers using OTP-based solutions [182–184], as well as PUF-based solutions [185, 186]. *Daniel Kelly et al.* [182] claim that IoT devices with single-factor authentication are not sufficient for secure communication. A solution presented by *Shivraj et al.* [183] creates a lightweight, robust, and scalable OTP technique developed by using the principles of Identity Based Encryption (IBE)-Elliptic Curves Cryptography (ECC) allowing two-factor authentication. The work done in [184] also has an authentication through OTP to the application level information security.

2.4 Network Traffic Security Solutions

This section overviews some techniques for network traffic control, including the selection of rules for blocking traffic.

Most firewalls have blocking rules that deal primarily with IP addresses. The relationship between a domain and its IP addresses is slightly loose because a domain can have multiple IP addresses that can change frequently. For example, in the *iptables* tool, it is more complex to block any connection from a full domain name, including all subdomains, because it requires a reverse lookup to see the mapping of an IP in the domain. If it is HTTP(s), the request must be analyzed and the sub-domain determined. Besides this complexity, the tool creates a rule with the first IP address, and if the address changes, the *iptables* rules will be out of date.

Another problem is that an IP address can host many domains and, if users block or allow one of these IP addresses, all domains hosted on it will also be allowed/denied. Likewise, when we allow an IP that hosts many domains, we allow multiple services and create a security problem due to exposure to other types of services. An example is *Cloudflare* [187], which provides a large amount of services to websites and sits between the public Internet and a server. *Cloudflare* users do not point their DNS to their server; instead, the users point their DNS to *Cloudflare* and then point *Cloudflare* to their server. Therefore, the same IP address is associated with millions of servers (*Cloudflare*'s IP addresses). Therefore, when one of these IP addresses is blocked/allowed, all sites pointed to them are also blocked/allowed.

Proxy servers are another widely used method of blocking access to websites. However, without an SSL introspection mechanism, a proxy generally can not decrypt HyperText Transfer Protocol Secure (HTTPS) and, therefore, can not know the IP address and content. To drop the connection, we would need at least information about the IP address/URL. For this setup, we tested Squid [188], which has an option to decrypt requests transmitted using the HTTPS protocol called SSL Bump. The solutions based on a proxy remains one of the most effective to solve the problem. However, SSL bump requires a man-in-the-middle to intercept communications to get the encrypted data and the IP address/URL information we need to drop the connection, which would make us susceptible to MiTM. Along with this, decryption of HTTPS tunnels without consent may violate ethical standards and be illegal, depending on the jurisdiction [189]. From a practical point of view, in this scenario, we have data similar to an Internet Service Provider (ISP); if we use the proxy, no communication will be secret, with all messages exchanged in *WhatsApp* or *Google*, for example, exposed on the local and remote machine.

With Autonomous System Number (ASN), we can block specific domains per network operator, but the opposite is also true. For example, if we only allow Google, we are also allowing all advertising domains associated with it and entities controlled by that network.

Another method to block specific traffic can be based on the Server Name Indication (SNI) [190]. By default, TLS [191] connections do not provide a client's mechanism to discover the server it is contacting. The header containing the Fully Qualified Domain Name (FQDN) [192] is encrypted, so we cannot access that name. To overcome this problem, it is possible to use the SNI extension for TLS, where a client indicates which hostname it is attempting to connect to at the beginning of the handshaking process. This field is part of the *ClientHello* message of the TLS negotiation (before encrypted communications) so that we can access the unencrypted hostname. The main advantage of choosing a firewall solution based on the SNI is that we can allow/block a specific hostname connection by solving the problems mentioned by the other solutions we tested.

For the implementation of this concept, we use an extension for *iptables* to filter TLS traffic based on the SNI [193]. In addition, we use a combination of *SNIdump* (similar to TCPdump) and *whois*, to provide the user with real-time information. SNI also constitutes some impersonation problems [194]. However, a device will not accept the connection with itself if the certificate does not match the desired one, so it is not a problem. Also, with *iptables*, the user can define a permission that takes effect only during a specific time window so that there are no unwanted communications outside the period defined by the user. For an attacker, it would be challenging to impersonate the certificate precisely during the period in which the user allows communication, in addition to needing to know which devices the user has.

However, with the introduction of TLS 1.3, DNS over HTTPS (DoH) and DNS over TLS (DoT) can encrypt DNS queries, and when combined, TLS 1.3 and Encrypted Server Name Indication (ESNI) can also prevent SNI leaks. Despite the advantages of blockchain traffic with SNI, this technology leads to a privacy gap because, on establishing an encrypted connection with the correct credentials, SNI transmits the website's domain name in plain text, which means that a passive attacker can gather the server's name and track the visits. According to the Internet Engineering Task Force (IEFT) draft [195], there is a solution to this security issues, which is called ESNI, which consists of a methodology to sent the SNI field encrypted in the first HTTPS packet. The idea behind ESNI is to prevent TLS from leaking any data by encrypting all messages while maintaining connection privacy. ESNI implies TLS 1.3, so that the certificate and embedded hostnames are encrypted. With ESNI enabled, and using secure DNS transport, such as DoH or DoT, the server name will not be visible on the connection. ESNI helps preventing this by masking the server's name during SNI, which means that although the ISP can view the connection, it cannot see which domain the user is trying to access. Therefore, it is very likely that SNI will stop working, and we have to keep up with this transition of TLS 1.3. So, this is a great limitation for the *iptables* solution. Also, in its traditional form, the DNS protocol allows the interception of domain names that the user browses, as it lacks any encryption since its initial definition. One of the steps to be taken to protect privacy on the Internet is to switch to a new version of DNS servers that allow encrypted access via TLS 1.3 or HTTPS. In this way, it will be possible to hide domain resolutions and cover this first breach of privacy.

In addition to these mechanisms, we also analyzed some academic and business works that implement traffic control, focusing on IoT solutions.

FANE [196] is a firewall between the device and the network to generate and enforce traffic rules autonomously. It tries to block what is not essential for operations. The system uses firewall rules to drop all packets that are not allowed by generated rules. The authors claim that IoT network segments must be segregated from the home network where there are sensitive tasks (e.g., bank accounts). *FANE* learns the rules during operation (network traffic) generated while it is not compromised.

From the evaluations done by the authors, only one device was not working properly after *FANE* has activated its firewall rules due to a specific load balancer. However, this problem could be solved by accepting IP addresses close to addresses that *FANE* already knows. For us, this last part is already a limitation. *Daniel Amkær Sørensen et al.* [197] also propose a system that generates rules based on the real-time traffic analysis.

Security and Privacy for In-home Networks (SPIN) [198] is an anomaly detection module implemented in SIDN Labs. The authors promise to do traffic inspection and make automatic and personalized blocks by the user. In their work, the authors promise to block Distributed Denial of Service (DDoS) derived from malicious devices on the network and allow the user to block traffic. Blocking is done based on patterns (unusual behavior), lists, e.g., *Snort* [199], or customized by the user. Another technique for semi-automation blocking is based on the purpose of *Anna Maria Mandalari et al.* [200] that automatically detects if such destinations are critical, i.e., required for proper device functionality, and iterative filtering the non-critical ones. The technique used is DNS override.

Pi-hole [201] acts as a DNS sinkhole, providing a fake IP address when there is an IP request for known ad-trackers. The difference between this system and DNS-based blacklist providers, such as *OpenDNS* [202], is that the DNS server runs locally on the RP3, which inherently gives greater control and therefore privacy.

2.5 Data Privacy Systems for IoT

This thesis focuses on data privacy techniques. This section overviews some data privacy systems for IoT, especially focus on data sharing control through configurations or privacy policies.

There are already some practical proposals for limiting data sharing, but more focused on other platforms like Android. An example of this is *ipShield* [203], where the authors propose an ”inference firewall” that basically what it does is to allow fitness data to be shared, but sensitive data, such as location, password, media, and psychological habits, are not shared.

Anupam Das et al. [204] want to create an IoT standard that supports the discovery of nearby IoT resources and how the resources collect and use data. It also allows configurations of what these resources can expose to users, such as opt-out configurations/opt-in. They assume that it is unrealistic to think that we will overnight make an Alexa device turn off when the children arrive, but that with the introduction of GDPR, research and work should begin.

More recently, *Olha Drozd and Sabrina Kirrane* [205] contribute with a Consent Request User Interface (UI), giving users the ability to make a complete customization to control their consent specifically to their wishes. This system offers users the possibility to grant permission to process only specific data categories for chosen purposes. Users can grant permission to process their heart rate and store it on their device without sharing it with anyone else. However, this paper only proposes the UI for the idea.

Abduljaleel Al-Hasnawi and Leszek Lilien [206] provide a solution that enforces privacy policies anywhere in the IoT. Hence, it is related to the data itself and identifies privacy threats, such as linkability, re-identification, non-repudiation, traceability, information disclosure, content unawareness, and non-compliance. This solution can be a complement to our solution, helping to detect sensitive information to be blocked.

Anupam Das et al. [207] propose a system that discovers the IoT devices nearby users and see the data privacy policies associated with these resources. It also allows users to discover configurable parameters (opt-in, opt-out, data erasure) to help users manage the IoT from privacy. As future challenges, authors are exploring machine learning models to build and refine user policies and preferences so that it is possible to configure data privacy according to policies and help users define those policies.

Vijay Sivaraman et al. [208] propose an architecture with three components: ISP, Security Management Provider (SMP) and Home Network. The SMP component is responsible for the security management of access controls. The authors build a prototype evaluation with some devices, including a Nest smoke alarm; in this case, when privacy protection is enabled, the system requests the SMP to make an API call to prevent the device from accessing the log servers (where it sends 250KB per day). Compared to our approach, the concept is different, and there is a limitation in the choice of permissions, as only developers can define what to block for each category (security or privacy), with no options for the regular user.

2.6 Summary

This chapter's goal was to search for a suitable solution to address our target system's requirements. The features include a device identity scheme that guarantees end-to-end communications included as a middleware solution component. The network controls should include traffic monitoring and local data (offline), and devices must have different "personas" with different behaviors depending on the device's real-time environment. The middleware should be adapted to integrate on "closed" devices (that it is impossible to modify the firmware) and devices such as Raspberry PIs.

As no solution integrates all of this, we focused on systems that belong to the intersecting domains, namely, authentication and secure communications, network traffic security solution (that controls data sharing), and data privacy middleware systems, to see if we could extend one of them and avoid designing and implementing new solutions from scratch.

Using the insights learned from several inspirational middleware systems and authentication schemes, we have designed a middleware that includes the following features:

- Device identity based on a hardware solution that stores a PKI offline;
- Implementing authentication to provide secure end-to-end communications;

- Users' privacy based on firewall-like solutions that provide full control of network data and configurable privacy settings for internal and external traffic data;

Also, the middleware is the basis for new features that we want to include in the future.

Chapter 3

End-to-End Secure Communications

"We cannot solve problems by using the same kind of thinking we used when we created them."

— Albert Einstein

One of this thesis's objectives is to ensure secure communications with end-to-end solutions so that devices can communicate and authenticate securely. Securely exchanging data and information is not new in the literature. PKI is one of the most used solutions for authentication on IoT [209]. However, many of them are unsuitable for IoT, and they depend on a centralized online entity representing a SPOF. Human error or hacking can compromise a PKI and require permanent connectivity even with fault tolerance and resilience.

To address the constraints as mentioned earlier in the description of the current solutions (section 2.3), we have developed a new decentralized solution [210] based on the ZRTP [169] protocol, which is a key agreement protocol used in setting up a secure call in VoIP. We present two new proposals, namely KEAV [211] and pTASC [212], that allow end-to-end secret key negotiation without requiring the participation of any other device or human interaction after the first communication/iteration. We evaluated these protocols to see the feasibility and security regarding IoT, and compared them with PKI.

3.1 KEAV Overview

The purpose of KEAV focus on achieving secure end-to-end communications, enabling devices to exchange data privately. It uses a key exchange DH to overcome the complexity of PKI systems or any trusted third party. However, DH alone cannot guarantee authentication, and has a MiTM problem [213]. *Zimmermann P. et al.* [169] proposed a solution that allows the detection of MiTM attacks by displaying a SAS for users to read and compare verbally over the phone at VoIP communications.

The authentication of users is intrinsically linked to users' voice on the comparison of the SAS. The first obstacle in adapting ZRTP for data is due to the impossibility of comparing SAS by voice.

For the IoT, we decided to create an extra layer that automatically compares the SAS. KEAV uses MPC to compare the SAS automatically, privately, and without human intervention. Even for VoIP communications, this layer is important because this double-check is critical for some users who, by mistake, forget to validate the key.

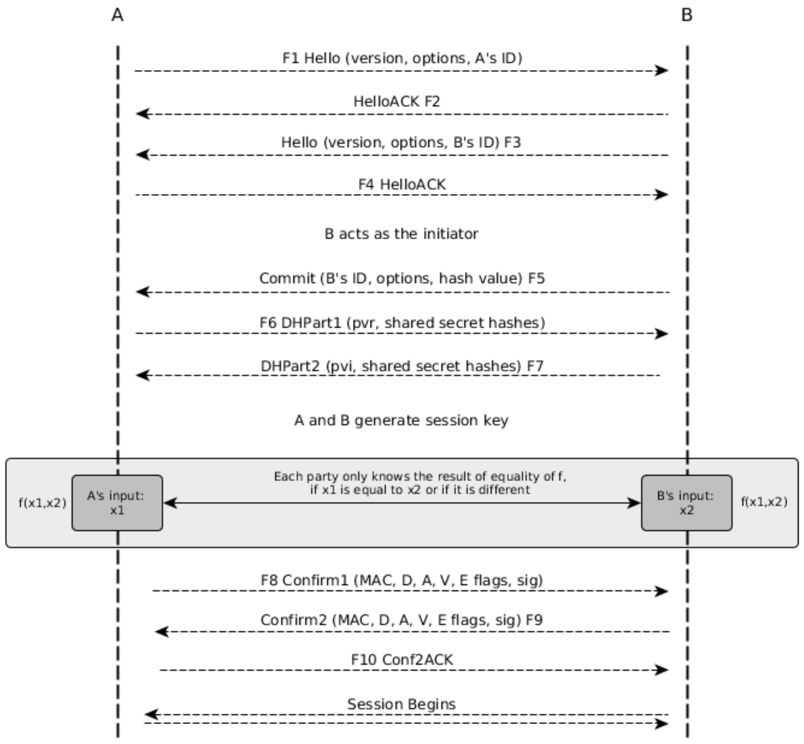
A
B
F1 Hello (version, options, A's ID)
HelloACK F2
Hello (version, options, B's ID) F3
F4 HelloACK
B acts as the initiator
Commit (B's ID, options, hash value) F5
F6 DHPart1 (pvr, shared secret hashes)
DHPart2 (pvi, shared secret hashes) F7
A and B generate session key
f(x1,x2)
A's input: x1
Each party only knows the result of equality of f, if x1 is equal to x2 or if it is different
B's input: x2
f(x1,x2)
F8 Confirm1 (MAC, D, A, V, E flags, sig)
Confirm2 (MAC, D, A, V, E flags, sig) F9
F10 Conf2ACK
Session Begins

Figure 3.1: KEAV overview

Figure 3.1 shows the communication scheme between two devices (A and B) through KEAV. This scheme improves the Request For Comments (RFC) [169], thereby eliminating the need to verbally compare a SAS through human manual verification. The protocol starts with both terminals exchanging messages *Hello* and *HelloACK*. Both devices' IDs are revealed and validated (Pseudo-Random Number Generator (PRNG)) on these messages. After this exchange, the key agreement's exchange can begin with the message *Commit*. The endpoint that sends the *Commit* message is considered the KEAV session's initiator and drives the key agreement exchange. Our approach inherits two main modes of agreement:

- DH mode: The DH public values are exchanged in the *DHPart1* and *DHPart2* messages. So, in this mode, KEAV endpoints exchange a new shared secret through the DH exchange;
- Preshared mode: In this mode, the DH calculation is omitted by the endpoints, as it assumes that there is a known shared secret from a previous session.

After this phase, the MPC component is available, which automatically validates the SAS. It is then possible to send a *Confirm* message from both the devices indicating that they have accepted and verified the SAS. Finally, the protocol is ready to start sending data in this secure session.

KEAV takes advantage of some security and privacy properties, such as forward secrecy, to ensure confidential communication in the future, which means that when a user makes multiple connections to a client, the protocol turns the keys. In this way, all communication keys are different. At that point, an identifier file caches the symmetric key material used to calculate the session's secret keys, and these values change with each session. If an attacker gains access to the local seed to derive future and current keys, he will not reproduce any previous information exchange. In other words, if a single communication is compromised (the session key "was leaked"), it does not compromise the confidentiality of all previous communications. In this way, KEAV can guarantee additional communication protection because, even if users do not care about SAS, there is still reasonably decent authentication against a MiTM attack, based on a form of key continuity.

A key negotiation generates a sequence of letters and numbers whose size can be defined and equal at both ends. In the case of ZRTP, any attempt to capture and decipher the voice will cause the strings to be different (a string is a mathematical function derived from the primary key at both ends, so any difference in the key will generate different values). The key continuity is often confused with forward secrecy. However, the idea of key continuity focuses on using a key that is limited to one use. ZRTP uses the old key to generate the new one, which also means that the key increases complexity with each use.

For now, this solution assumes that there is a secure end-to-end authenticated channel to work securely.

3.1.1 Operation Modes

We have created three different possible modes for this system (Table 3.1) so that users can choose the one that best suits the situation in question. There are "paranoid" ways to check security when necessary for each iteration, for example, on anonymous or sensitive data issues, and these modes will be more complex to ensure an extra security check. We will have other equally safe modes that do not have extra security checks but generate SAS for some iterations and rely on the key's continuity.

Without Key Continuity Mode

The first mode of use is entitled the "Without Key Continuity Mode." This mode takes advantage of the existing mechanism of the ZRTP, presented in section 4.9.1 of the ZRTP draft [214], where the authors suggest cache-less implementation. This feature will always allow the KEAV to be executed in DH mode (also based on the ZRTP DH mode [214]). Nevertheless, it requires to compare the SAS with the MPC.

Modes	**MPC**	**Key Continuity (KC)**
Without KC Mode	●	○
KEAV Normal Mode	Every 10 it.	●
KEAV Paranoid Mode	●	●

●Included;

○Not Included.

Table 3.1: KEAV - Execution modes properties

This method presents itself in a safe state with some security issues solved, including the impossibility of an opponent being able to obtain the local shared secret.

In brief, users must validate the keys in all sessions because they never agree on a key beforehand. There is no need for extra storage on this operation mode, which decreases the overhead on the network and on the device itself.

KEAV normal mode

The KEAV normal mode is based on the existing mechanism of the ZRTP, presented in section 4.4.2 of the ZRTP RFC [214] and denote the pre-shared mode.

When the normal mode uses a key continuity feature, it is unnecessary to compare the SAS in all iterations with MPC. However, this check happens every *N* connection (the user can define *N* - the default is 10) to verify that the protocol is working as expected.

This mode preserves the forward secrecy and key continuity properties.

KEAV paranoid mode

This mode also uses key continuity as the normal mode. However, this protocol is ”paranoid” as it requires extra verification in all iterations to ensure no errors in the KEAV protocol. In this mode, the MPC is used on all connections to compare the SAS.

This mode requires a more significant overhead to ensure an extra safety check. It is ideal for more sensitive data because it is possible to guarantee that the protocol is working correctly in all connections and that the SAS is equal in the two connected devices.

3.1.2 Evaluation

This section presents the results of the proposed system in terms of network latency. We performed experiments in all implemented modes (described in Section 3.1.1): without key continuity mode, KEAV normal mode, and finally, in KEAV paranoid mode. These modes differ in usage and complexity. With key continuity, KEAV not requires to use MPC in all iterations, but on the other hand, in some cases, it may be essential to validate the SAS at every iteration to ensure that the protocol is always secure. Therefore, it is crucial to analyze whether there are significant differences in latency.

We also compare the runtime of KEAV versus the OpenSSL PKI system. In PKI systems, the trust is built based on public-key certificates [*]. So, trust depends only on the CA that digitally signs the certificate.

We are interested in measuring latency over the network in both systems. We use *clock* or *chrono* timers to measure the time that it takes to complete a particular action, in this case, the communication between the server (receiver) and the client (sender). If the client sends data to the server and waits for a reply, then the overall duration can be measured, which should roughly estimate the Round Trip Time (RTT) [†] between the client and server. We are mainly interested in measuring the provision time of the KEAV system vs. the PKI-based system. We also compare the running time of KEAV with and without MPC to assess the whole order of magnitude of MPC and its contribution to the overall latency.

This section is divided into four sections. The first section consists of the description of the setup and configuration of the system for the experiments. Then, we present the implementation details, followed by evaluating the different modes that we implement. Finally, we present a discussion on the different results between PKI-based systems and KEAV.

3.1.2.1 Setup

KEAV aims to integrate into an IoT environment. The experiments used two common off-the-shelf Raspberry PI 3 (RP3) Model B running the Canonical's Ubuntu Core, a specialized operating system for the IoT, to measure the results of this system's usage in a realistic environment (low-resource devices).

As, in its essence, IoT emerges from the interconnection of devices, we decided to create an environment as shown in Figure 3.2, where two RP3 connect over the Internet, with one end connected using a wireless link through a wireless router (ASUS RT-AC3200), and the other end connected using a wired Ethernet link. We decided to route the traffic through the Internet to measure execution time with network latency in a realistic scenario.

3.1.2.2 Implementation

The implementation of this system is based on two libraries: *ZRTPCPP* [215], which is a C++ implementation of the ZRTP protocol, and *ABY* [82] library, which is an MPC framework (in this

[*] A public key certificate is a data structure that associates a public key with a given agent (a representation of its identity).

[†] In the context of these types of systems, this is the time it takes for a data packet to travel from a specific source to a specific destination and back again.

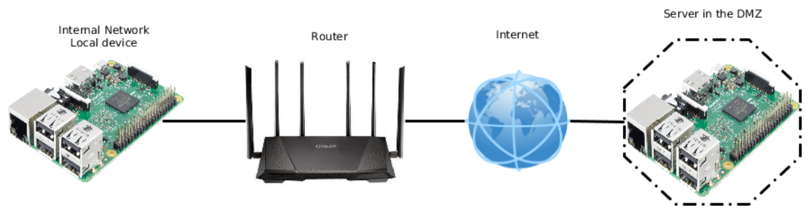

Figure 3.2: KEAV - Setup for performance experiments

case, focused on secure two-party computation) [216].

ZRTPCPP

We decided to construct the new protocol based on the *ZRTP* for end-to-end communication because of the support given by the development community and the existence of a complete ZRTP implementation with well-known libraries as dependencies. For this reason, we choose the library *ZRTPCPP*.

ABY

ABY combines secure computation schemes based on Arithmetic sharing, Boolean sharing, and Yao's garbled circuits.

The ABY library includes some sample applications, such as the millionaire's problem, secure-computation AES and euclidean distances [82]. We create a module that obtains the local file's input and implements the equality problem (similar to the millionaire's problem).

For this adaptation, we build an equality problem circuit with the function $out = bc- > PutEQGate(s\_alice, s\_bob);$, and we add a new parameter to the function $test\_equality\_prob\_circuit$ to pass the SAS to the function by parameter within the *showSAS* function of the ZRTP.

In this system, MPC has only the function of comparing the SAS and guaranteeing that the exchanged SAS by the protocol is correct.

Integration

In terms of both libraries' implementation and setup, we start by match both users (sender and receiver). Thus, we perform the integration between the two libraries; namely, the receiver of the ZRTPCPP became party 0 of the MPC, and the sender became party 1.

We did the integration of both libraries using *cmake*. In terms of execution, when ZRTP makes the SAS display, we add a new layer of verification of this SAS. On the other hand, we call the ABY framework up to verify the equality of the SAS.

In this implementation, there are two schemes: when the SAS is equal (verified by MPC), then the data can be exchanged securely and privately, otherwise if the SAS is different (also verified by

MPC), the protocol aborts without any exchange of data between the two parties. When the SAS is different, there is probably a MiTM attack in the communication.

3.1.2.3 Results

In order to evaluate KEAV, we measured the provisioning times in all three modes. Figure 3.3 depicts the scenario to be evaluated, representing the provisioning phrase of KEAV.

Key Exchange
Client
Server

Figure 3.3: Provisioning of KEAV

To perform these measurements, we used the *gettimeofday()* system call, using the setup described in Section 3.1.2.1. For each mode represented in Table 3.2, and the corresponding iteration from one up to ten, these values are the average values after ten repetitions and the corresponding standard deviation of the average.

Modes	**Iter.**	**Without MPC (ms)**	**With MPC (ms)**	**KEAV (ms)**
Without KC	[1-10]	290 ± 2.27	9553 ± 49.36	9843 ± 50.49
KEAV Normal	1	288 ± 2.27	9457 ± 49.36	9745 ± 50.49
	[2-9]	283 ± 1.78	-	**283** ± 46.92
	10	287 ± 1.78	9495 ± 45.28	9783 ± 46.92
KEAV Paranoid	[1-10]	291 ± 2.27	9506 ± 49.36	9798 ± 50.49

Table 3.2: KEAV - Latency of all the three modes presented in the section 3.1.1

Although the ”Without Key Continuity” mode does not require extra storage because it is unnecessary to store the previously exchanged keys, this mode always takes an average time of approximately $9843ms \pm 50.49$ in all iterations. This mode has some advantages regarding security (it never relies on previously exchanged keys and has key continuity) and storage. However, the running time is the highest of the three modes.

The ”KEAV normal” mode, unlike the previous model, uses key continuity and, as such, requires extra storage for the key. However, KEAV uses MPC every ten iterations. In this case, the running time is $283ms \pm 46.92$ after the first iteration. In the first iteration, given the overhead of MPC, the latency is $9745ms \pm 50.49$. As we use MPC at every ten iterations, it originates from a latency spike.

The ”KEAV paranoid” mode is similar to the ”normal” mode, as it also uses key continuity. However, this protocol is ”paranoid” to verify the SAS through MPC in all iterations, thus not relying only on key continuity.

There is no difference between the ”Without Key Continuity” and ”KEAV paranoid” modes in latency. Only the KEAV ”normal” mode has a significantly lower latency than the other two modes; notice that it runs MPC every ten iterations. All the other iterations have a similar behavior. On the upside, the average running time for the remaining iterations is 283 ± 46.92.

In this mode, the main drawback is the additional required storage. The implicit trade-off is between storage and latency.

The choice between the operation modes depends on the type of devices used, the communication links between them, and, ultimately, the user’s functional requirements.

As an authenticated channel is assumed, we added a δ to the runtime to represent the overhead associated with it. However, we assume that this δ will represent more than 20% of the runtime.

3.1.2.4 PKI vs. KEAV

Figure 3.4 shows the PKI scenario to be evaluated. Here, we have a client, represented by a local RP3, and a server hosted in a remote RP3 (illustrated in Figure 3.2) provisioned with DigiCert certificates. The remote certificates support the Online Certificate Status Protocol (OCSP) stapling that removes customers’ complexity communicating with the CA. The TLS server periodically requests the OCSP responder about the validity of its certificate and caches the response. The OCSP responder returns an OCSP response, which is (directly or indirectly) signed by the CA that issued the certificates. The TLS client can treat this stapled OCSP response in the same way, i.e., the certificate needs to have a valid timestamp and signature to be used.

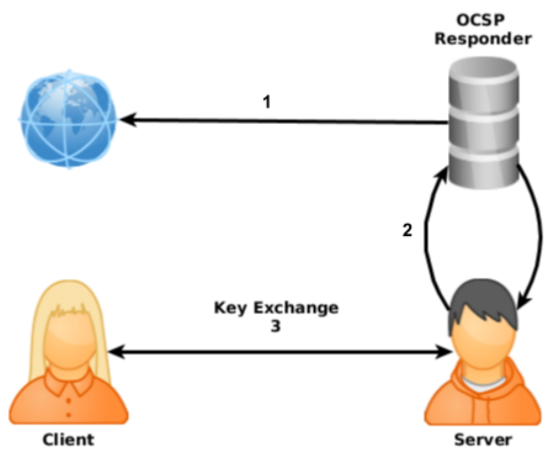

Figure 3.4: Provisioning with PKI

During the TLS handshake, the client announces support for OCSP stapling to the server. In turn, the server activates the Certificate Status flag if it has supports for it. Figure 3.4 shows this process in step 1. During step 2, Alice and Bob establish an SSL connection. The goal is to measure the certificate's OCSP (state) check latency on top of the latency added by the SSL connection handshake.

To measure the latency of this scenario, we used *s_client* and *s_server* binaries from the OpenSSL library. We ran *s_server* service on the server machine and made the *s_client* run on the local machine so that we could obtain both the OCSP information and server/client handshake results. We obtained an average running time of $380ms \pm 11.60$ across the ten iterations.

Our system running the "normal" mode achieves a latency reduction of 26% compared to the results using PKI-based systems.

3.2 pTASC Overview

However, on KEAV, we found a security problem. During the security analysis, we found that the MPC does not provide authentication. Thus, without any Session Initiation Protocol (SIP) channel that authenticates the two parties, KEAV is subject to a MiTM during the exchange of the SAS. Along with this, the current implementations of MPC are not secure implemented, as stated by *Resende et al.* [92].

To overcome this issue, we decided not to change the MPC layer, for now, until there is an authentication solution for MPC and secure implementations.

Thus, we use an adaptation of the LLC concept, already described in the section 2.3.2. This concept aims to create an extra channel (secure and authenticated channel) to securely exchange the SAS without being vulnerable to a MiTM attack. Then, the SAS is compared locally on both client sides. For the exchange, we use infrared as a LLC. As infrared is more limited in terms of range, we can ensure that devices wanting to exchange information have to be nearby. This paradigm is valid in two scenarios: all participants are in the same room, and all the participants trust each other *a priori*.

The difference between KEAV and pTASC is the comparison component as we switched from MPC to the infrared component. pTASC uses infrared to exchange SAS privately, and the comparison is made locally on both sides. It is then possible to send a message *Confirm* from both devices, indicating that both have accepted and verified the SAS. Finally, the protocol is ready to start sending data in this secure session.

3.2.1 Evaluation

This section shows the performance tests regarding the SAS comparison. The provisioning time evaluation is provided in the section 3.1.2.3 and the section 3.1.2.4. These results are equal because the ZRTP implementation is the same, and the provisioning phase is independent of the SAS comparison method.

3.2.1.1 Setup

To test the operation of the protocol in the real world, we create an IoT environment, as shown in Figure 3.6, where two RP3 Model B, running *Raspbian Stretch* [217], are connected over an ad hoc network. An Infrared (IR) Transmitter (KY-005) and an IR Receiver (KY-022) are used as infrared sensors to exchange the SAS key.

3.2.1.2 Implementation

In this section, we describe the implementation of DTP (Data Transport Packet) which is an adaptation that we made for ZRTPCPP.

Regarding the comparison layer, we implement the Infrared with *pigpio* [*].

ZRTPCPP
The *ZRTPCPP* implementation is deeply connected to *GNU ccRTP* which is the software responsible for accept connections, differentiate messages from different protocols, and pass them to the respective handler. The Real-time Transport Protocol (RTP) protocol uses User Datagram Protocol (UDP) protocol, so, at the application level, there are mechanisms to ensure the reliability and filter out-of-order packets. To separate these two packages, we create a package called *DTP*, that emulates the *GNU ccRTP* to facilitate the connection with the *ZrtpQueue*.

In comparison with *GNU ccRTP*, the *DTP* differs in the transport protocol, because it uses Transmission Control Protocol (TCP) instead of UDP. We choose TCP because the protocol we are developing is to data, so if a packet is lost, it must be re-transmitted. As we use TCP, we do not need to implement a reliability mechanism at the application level (as it happens on the RTP) because TCP already provides reliability mechanisms implemented [219].

The *DTP* package needs to be able to receive and manage many connections simultaneously. Without this capability, the protocol would communicate with only one device, which negatively impacted performance when a device needs to talk with several peers. This type of necessity is common to servers, where there are two main architecture types of servers: multiplex [220] and multithread [221] servers. The multiplex server [220] is an event-driven approach that uses asynchronous I/O, and a unique process is responsible for multiple connections. The server changes between the connections when an event exists to process. The multithread [221] approach associates each incoming connection with a separate thread, where synchronous blocking I/O is the natural way of dealing with I/O. Given these two options, we opted for the multiplex server because of the low memory capacity of an IoT device, which is not compatible with a multithread server. As we needed

[*] *pigpio* [218] is a library for the Raspberry, which allows control of the General Purpose Input Outputs (GPIO) library

to implement a multiplex server, it was necessary to find a way to implement the non-blocking I/O. We use the native function *select* to implement the necessary asynchronous mechanisms, allowing a program to monitor multiple file descriptors, waiting until one or more of the file descriptors become ”ready” for some class of I/O operation.

The *DTP* package is organized in four classes: *DTPServer*, *DTPHandler*, *DTPConnection* and *Worker* (Figure 3.7).

The *DTPServer* has the role of Facade class, allowing the programmer to define user callbacks, handlers for packets, and connections with other peers. The *DTPHandler* manages the connections and workers. This class contains a list of all connections and locks to manage the multiplex architecture. Furthermore, this class has the *get_next_socket* function, that monitors the different connections using the *select*. The *Worker* class is the base for the multiplex server; it has his thread and will process which of the events identified by the *DTPHandler*.

Each one of the connections has an instance of the *DTPConnection* associated. This instance keeps the context of the connection. This context has different pools of packages, from the cryptographic context to the communication channels. This type of structure is equivalent to the *GNU ccRTP*. The *DTPConnection* is the class responsible for the connection with the *ZrtpQueue* and each *DTPConnection* has one instance of a *ZrtpQueue*.

To structure the packages of the *DTP*, we decided to use a ZRTP packet as a starting point. The ZRTP packet has field constant on all ZRTP packets, called *ZRTP magic number*, which simply serves to identify the protocol. We use this approach to structure the *DTP* packets. We kept the same header structure of a *ZRTP packet*, and changed *ZRTP magic number* value to differentiate protocols. With this type of structure, we can add more protocols sharing the same connection channel.

After the implementation of the *DTP* package and after trying to join the *GNU ZRTP CORE* with the *DTP*, we realized that, on the *GNU ZRTP CORE*, there were dependencies related to *GNU ccRTP*, on functions involved on threads and thread concurrent synchronization. These dependencies problems were possible to solve using *Commoncpp* [222], a small library which is a dependent of *GNU ccRTP*.

Beyond the *DTP* implementation, we need to create a system that uses the SRTP keys generated on the ZRTP to transmit information safely on the same channel. Similar to *GNU ccRTP*, we associate a crypto context to each communication channel and cipher the packets on the channel using AES-Cipher Block Chaining (CBC). This algorithm’s use was due to an implementation on the *GNU ZRTP* CORE and so, do not increase redundant code.

Our protocol can be expanded by a programmer and integrated into an application. We allowed the developer to override the functions that process and receive data packages and all user callbacks hooked to the ZRTP protocol.

With all of the above components, we were able to build a new protocol to the IoT world that offers a tangible form to share information securely in a peer-to-peer way, without the participation of any other device besides the sender and the receiver.

3.2.1.3 Results

With the previous setup, we made a test to evaluate the secure session establishment runtime.

We measure the runtime during a SAS exchange. As we have an IR Receiver and an IR Transmitter, we make two exchanges from Alice to Bob and simultaneously, from Bob to Alice. We collected ten-time samples of one of the communications, with an average of time exchanging $0.19ms \pm 0.04$.

Finally, to establish the secure session, we measure the runtime from the start of the protocol until creating a secure channel, using the function *gettimeofday* from the native library *sys/time.h*. We collected ten-time samples, with an average time to connect of $1050ms \pm 58.5$.

In all, we have a mean runtime of $1050.2ms$ on the first connection.

3.2.1.4 Discussion

The runtime of our system increases 64% compared to PKI systems (we evaluate a PKI scenario in the Section 3.1.2.4). The use of an infrared channel behaving as LLC does not add much overhead to the overall runtime, meaning that the time of $1050ms \pm 58.5$ is the approximate base time. If we change infrared channel to another method, such as Near Field Communication (NFC), we have a time of $1050ms \pm 58.5 + \delta$, the δ being the execution time of the NFC.

This inefficient scenario is related to the change for TCP connections and the packet loss from the ad hoc network. Also, the use of PKI was tested based on a high-density certificate network, meaning that the runtime is essential, and everything is optimized to perform in the shortest time possible, compared to our implementations that are deployed in a local scenario and does not use optimized solutions.

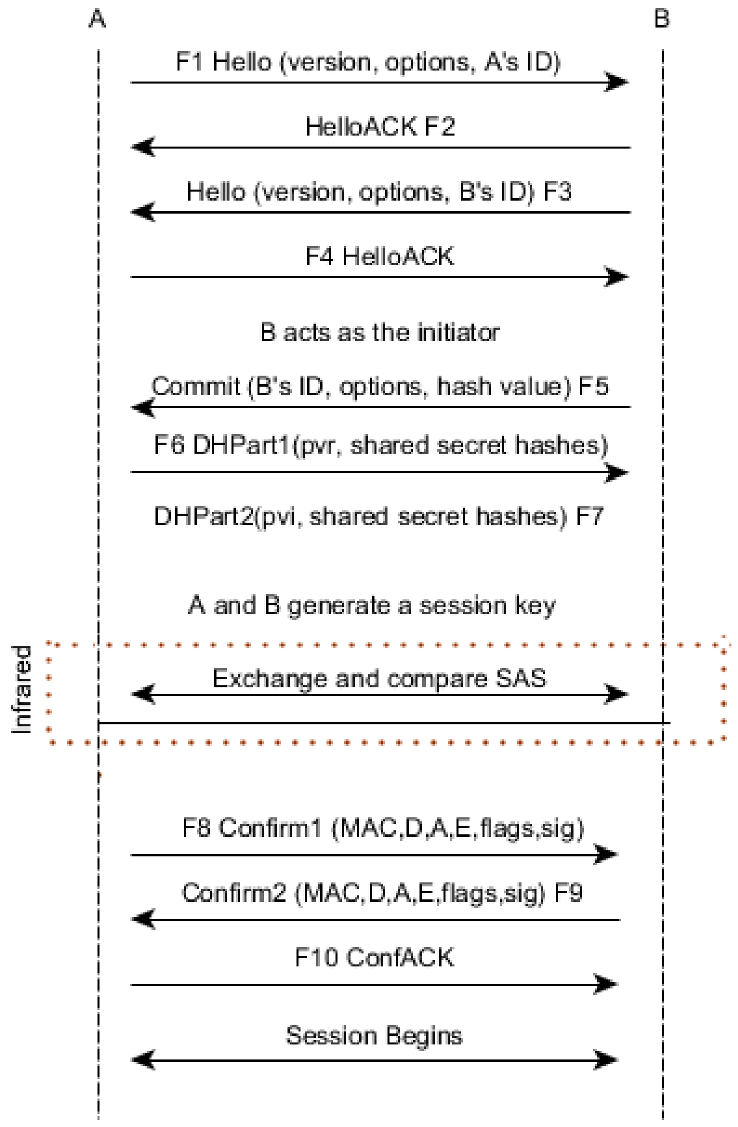
A
B
F1 Hello (version, options, A's ID)
HelloACK F2
Hello (version, options, B's ID) F3
F4 HelloACK
B acts as the initiator
Commit (B's ID, options, hash value) F5
F6 DHPart1(pvr, shared secret hashes)
DHPart2(pvi, shared secret hashes) F7
A and B generate a session key
Infrared
Exchange and compare SAS
F8 Confirm1 (MAC,D,A,E,flags,sig)
Confirm2 (MAC,D,A,E,flags,sig) F9
F10 ConfACK
Session Begins

Figure 3.5: pTASC overview

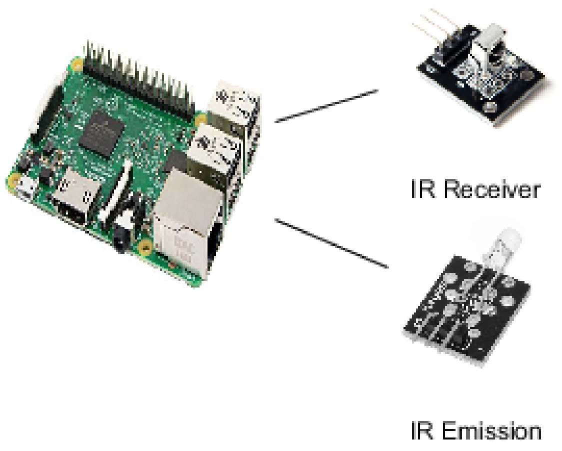

Figure 3.6: Ad-hoc network setup for pTASC evaluation

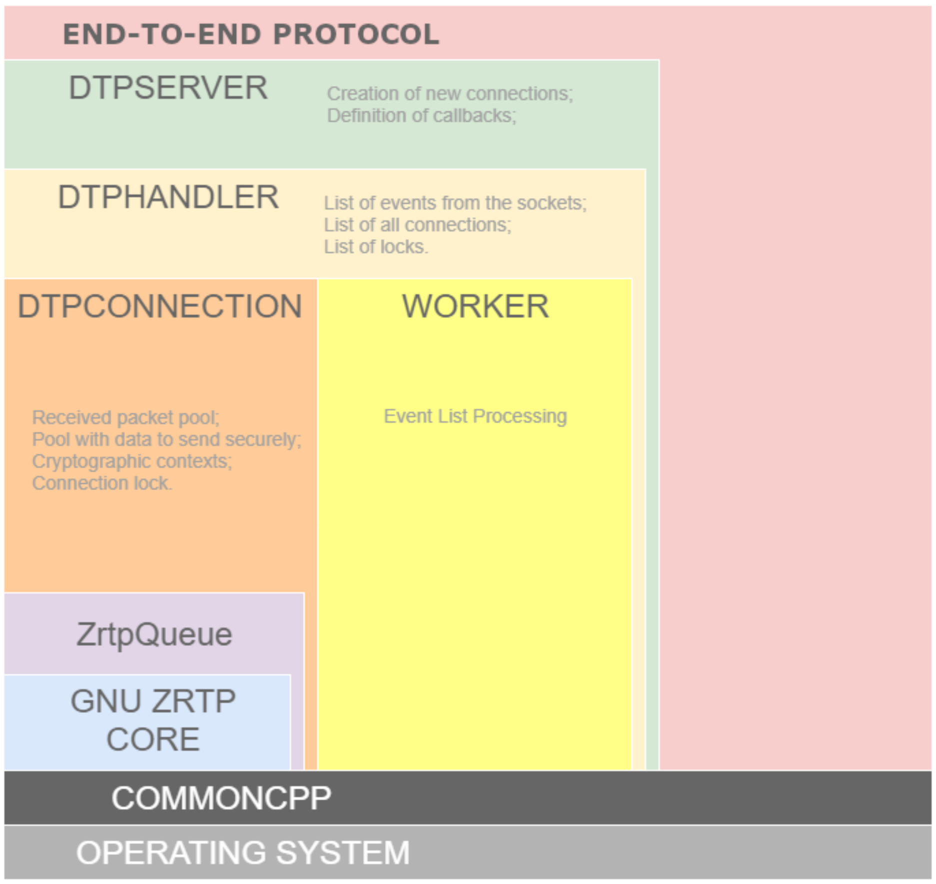

Figure 3.7: Relationship diagram of the DTP protocol implementation.

3.3 Security Analysis

This section defines a Threat Model, and we define some claims and the respective theoretical proofs for the claims.

3.3.1 Threat Model

It is possible for a collision of the SAS to occur, enabling a MiTM, i.e., an honest user can have a four-character size SAS key, and the attacker can get the same key and thus intercept the call through a MiTM [223]. However, increasing the SAS size can prevent this, as shown in the RFC of ZRTP [214], by increasing the cipher key size (the AES key), producing a new SAS with a more significant size. The increase in size adds security regarding the communication cipher and makes the SAS collision harder for an attacker.

Replay attack obtains information from one communication and tries to set the information in the next round. For example, if an entity A is exchanging a file with an entity B, an attacker called Mallory can send a piece of the previous file exchanged to try to corrupt the protocol. To solve this problem, KEAV and pTASC use the properties key continuity and forward secrecy to generate a new session key and make obsolete packets and ciphered information exchanged previously. These characteristics make the protocol secure against replay attacks because if an attacker performs this type of attack, A and B will ignore the information sent and continue the ongoing transfer.

João S. Resende et al. [224] presents traceability problems, where the attacker leaks information from VoIP metadata during the communication with the peers. Traceability issues are relevant in VoIP scenarios because there is no need to use this type of metadata when we trust a third-party server. In our scenario, we need to index the local cache information because we do not have information for other sources (peer-to-peer solution). This way, an attacker can obtain metadata information, but there is no solution to mitigate this metadata leakage while maintaining a scenario without a trusted third party, to the best of our knowledge.

To address the MiTM problem, pTASC was created with a LLC. However, attackers can try to intercept the communication and exchange a key with one of the devices. As we use infrared, it limits the attacker's distance to exchange data with the ”trustable” devices. This scenario assumes that all participants are in the same room, and all the participants trust each other *a priori*. Otherwise, KEAV cannot handle the first communication with MiTM attack because it needs an authenticated channel.

3.3.2 Attack Scenarios

In this subsection, we measure the security of the proposed solution against different scenarios of attack.

De-synchronization attacks.
The key to avoiding de-synchronization attacks is represented in the Figure 3.5, in which there is a unique identifier used to store information related to this client in a database. When the user sends the message F7, there is an update to the local cache, rotating the keys, i.e., when both parties have exchanged the information correctly, they update it. If, for example, the disk corrupts the information or another type of problem occurs, both devices, when performing the pre-shared mode, will not be capable of negotiating a session key and will drop to the first stage, where pTASC performs a new DH key exchange.

Tag impersonation attacks
The proposed protocol is secure against tag impersonation attacks based on the security provided by combining PRNG's and locally stored secret keys. Each device has one unique identifier (as stated in the figure 3.5 in F1 and F3) that each device uses for communication with any other device. If an attacker obtains device A's ID, he may attempt to send A's ID to a device B, impersonating device A, but he will not do it. As explained previously, we have an identifier file that caches symmetric key material used to compute secret session keys, and these values change with each session. Therefore, when the attacker tries to impersonate device A, he cannot go unnoticed by device B because, when he tries to generate a new key with the victim, the secret keys are not the same as his and, therefore, the device B interrupts the communication. It will have an authenticated channel in the first iteration, so it cannot complete the attack, regardless.

Replay attacks
A replay attack is an attack that obtains information from one communication and tries to set the information in the next round of the communication. For example, if A is exchanging a file with B, Mallory can send a piece of the previous file exchanged to try to corrupt the protocol. To solve this problem, KEAV and pTASC use the properties key continuity and forward secrecy to generate a new session key and make obsolete packets and ciphered information exchanged previously. These features make the protocol secure against replay attacks because if an attacker performs this type of attack, A and B will ignore the information sent and continue the ongoing transfer.

Resistant to Single Point of Failure.
In the case of PKI implementations, there is a CA that checks the certificates' validity. The CA establishes a link between public keys and identities of people or organizations. Therefore, customers have to rely on a third party.

The certification authority represents a SPOF, as once one is compromised, all peers are compromised as well. For example, *Let's Encrypt* has issued 15,270 ”PayPal” certificates [225] to sites used for phishing. A failure in this type of system compromises several entities.

Regarding both protocols, while facing an attack, it can only compromise at most one of the participants, not both, because it is a decentralized solution.

MiTM exploit SAS Attack
Martin Petraschek et al. [226] describe the MiTM attack on the DH alone and state that, in ZRTP, the authentication can be made by comparing the SAS with voice recognition, which is mandatory for the first connection and optional otherwise (as it guarantees forward secrecy). However, an attacker can try to imitate the voice of one party, deceiving the other party. As described in the document *"If the protocol does not use media encryption, Mallory simply forwards the RTP packets between Alice and Bob and can listen in to the conversation."* Also, recent studies [227] show that this is possible; for example, *Lyrebird* has a set of algorithms that clone anyone's voice by listening to just one minute of audio.

pTASC addresses the MiTM problem using Infrared as an additional layer of security. Contrarily, without identities, a MiTM attack cannot be prevented in KEAV because MPC cannot handle MiTM attacks [228]. However, the users' identities do not even need to be valid because of the probability of an attacker intercepting an MPC communication and, at the same time, being able to generate the information exchanged (SAS) equal to that of the other peer is very low.

Let α be the alphabet $\alpha = \{a, b, c, ..., z\}$ with size 26 and β be the numbers $\beta = \{0, 1, ..., 9\}$ with size 10.

$$\gamma = \alpha \cup \beta$$

The permutations calculate the probability of generating the same SAS of the other peer with repetitions with the formula: n^r. Assuming the default length of SAS (4) and the total number of characters (γ=36), we have $n = \gamma$ and $r = 4$, so, we have γ^4 possible cases.

Calculating the probability, we have only one possible cause for the attacker to generate the same SAS of the other peer. So, We have a probability of almost zero.

$$Probability = \frac{FavorableOutcomes}{PossibleOutcomes} = \frac{1}{\gamma^4} = 5.95e^{-7} \approx 0$$

If we increase the size of the SAS, the probability is even lower.

3.4 Summary

In this chapter, we start by proposing a combination with ZRTP and MPC to solve the secure end-to-end communications with MPC, making the SAS comparison private. However, the MPC still has authentication problems when using in IoT scenarios because it requires an authenticated channel that identifies the two parties.

So, to be able to test the security of the protocol, we use the concept of LLC to create an extra channel (secure and authenticated) to exchange SAS securely, without being vulnerable to a MiTM. Thus, SAS is compared locally on each side. As infrared is a more limited technology in terms of range, we have to guarantee that the devices that want to exchange information are close to each other, which is a safe paradigm that is valid in two scenarios: all participants are in the same room, and all participants trust each other *a priori*.

However, pTASC has the following limitations:

- Does not scale to the context of smart cities, or larger smart home environments;
- Requires a controlled environment;
- Nearby devices required during the provisioning process.

Chapter 4

Secure Provisioning for Achieving End-to-End Secure Communications

"If you put a key under the mat for the cops, a burglar can find it, too. Criminals are using every technology tool at their disposal to hack into people's accounts. If they know there is a key hidden somewhere, they will not stop until they find it"

— Tim Cook

The provisioning process is usually an arduous task that encompasses device configuration, including identity and key provisioning. Given the potential pool of devices, this process must be scalable. Furthermore, this task should be semi-autonomous to minimize erroneous configurations during this process. Human configuration errors are often the source of many security and privacy issues [229], and, to solve them, we need systems with better interfaces and tooling to help provision new devices.

One of this thesis's goals is to develop a provisioning solution that achieves: device identity, scalability, offline cryptographic assets, and resistance to MiTM. This chapter presents a novel approach for provisioning IoT devices that combines public-key cryptography algorithms with OTP inside a secure token. The secure token acts as an offline storage for private keys, allowing access to cryptographic operations to be kept offline without access to the network. Our solution uses a manager device that acts as an OTP Server that can be switch off to reduce the SPOF problem. This way, device identity is guaranteed by physical access to this physical token.

4.1 System Overview

This section describes all the architectural components to achieve decentralized, secure end-to-end communications. This description includes the provisioning phase, authentication mechanisms, and scalability extension.

4.1.1 Manager Setup Phase

A CA sub-system is represented by combining a secure token and a manager device. The secure token plays an essential role in a certification system, supporting the combination of OTP with Public Key Cryptography Standards (PKCS)#11 to create and manipulate cryptographic tokens [*]. In our system, we used a secure token to perform all the cryptographic operations inside it, ensuring that the private key never leaves the device to avoid the possibility of private key theft in a possible network intrusion. This device must be reliable and controlled only by trusted people, such as the network owner. All certificates signed by the device will be implicitly trusted. Currently, the systems that manage PKI require a high-security degree and are on an isolated machine.

As the secure token stores all the cryptographic keys, it eliminates the hassle of having all of the cryptographic assets on the managing device, which would lead to a SPOF. On the other hand, the manager device has an OTP Server that allows authentication through the use of a secure token, that is, ”something you have”, proving that the device that is attempting to authenticate owns the secure token. For this reason, anyone who has access to this secure token can authenticate with the manager. The OTP Server can be turned off when not in use, avoiding possible SPOF.

4.1.2 Device authentication

The authentication between a new device and the manager is needed to add new devices to the trusted device pool. We use a combination of public-key cryptography algorithms with OTP to authenticate a new device. In practical terms, the network owner inserts the secure token into the target device to add it to the trusted device pool. The Figure 4.1 shows the whole process.

This section will describe the cryptographic algorithm used in this proposal and then provide a detailed description of the entire process when provisioning new devices.

Cryptographic Algorithms
As cryptographic algorithms, we choose Elliptic Curve Digital Signature Algorithm (ECDSA) for signing and verification, and Elliptic-Curve Diffie-Hellman (ECDH) for encryption.

We choose ECC because it is better for low resource devices as ECC requires fewer resources and provides the same security level as Rivest-Shamir-Adleman (RSA) cryptography with a smaller key [230]. In brief, it is possible to use RSA, but as the focus of the approach is IoT, we choose

[*]The PKCS#11 standard/protocol is widely used by applications that use cryptographic operations with non-exportable keys, as the protocol defines a standardized specification for interaction with cryptographic hardware (Smartcards, Tokens and Hardware Security Module (HSM)s). It is an abstract layer to perform the separation of the keys from the operations, allowing them to perform operations on cryptographic objects, such as private keys, without requiring access to the objects.

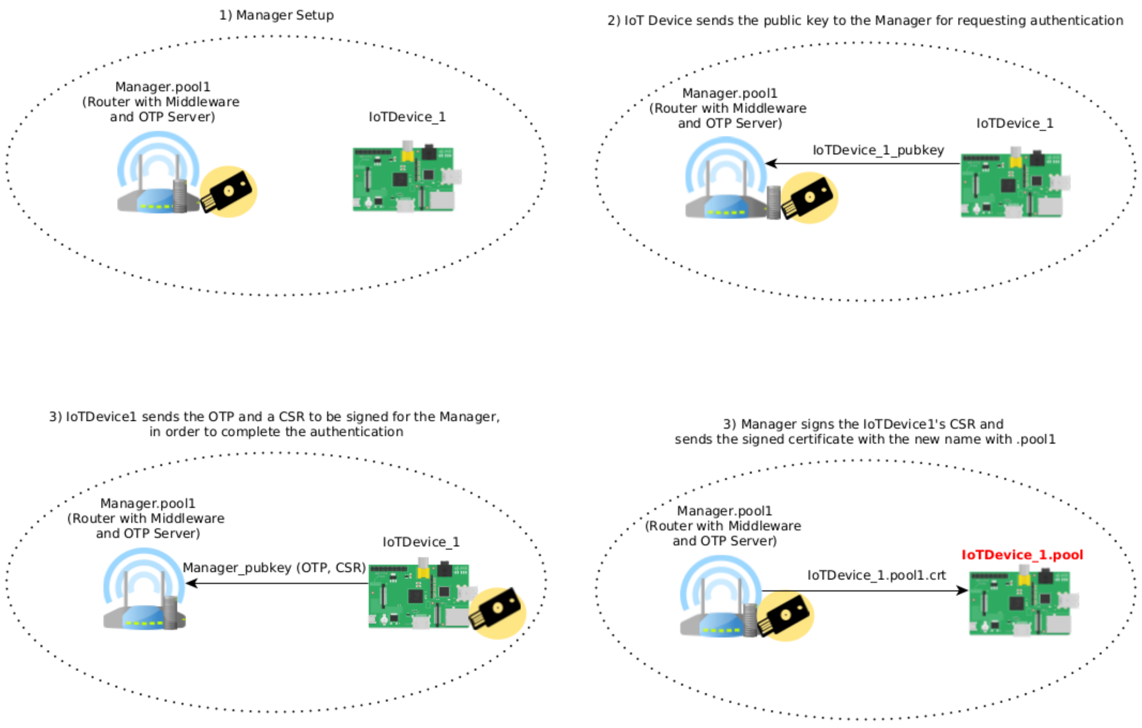
1) Manager Setup
Manager.pool1
(Router with Middleware
and OTP Server)
IoTDevice_1
2) IoT Device sends the public key to the Manager for requesting authentication
Manager.pool1
(Router with Middleware
and OTP Server)
IoTDevice_1
IoTDevice_1_pubkey
3) IoTDevice1 sends the OTP and a CSR to be signed for the Manager,
in order to complete the authentication
Manager.pool1
(Router with Middleware
and OTP Server)
IoTDevice_1
Manager_pubkey (OTP, CSR)
3) Manager signs the IoTDevice1's CSR and
sends the signed certificate with the new name with .pool1
Manager.pool1
(Router with Middleware
and OTP Server)
IoTDevice_1.pool
IoTDevice_1.pool1.crt

Figure 4.1: Device provisioning processes

ECC.

ECDH is a shared-secret derivation protocol that uses the elliptic curve form of the DH protocol. In this protocol, two parties can agree on a shared secret over an insecure channel using the knowledge of their own ”private key” and their partner’s ”public key” to generate a shared secret. Generally, the private keys are random numbers used for the key negotiation and then discarded (ephemeral). According to the *NIST SP 800-56A* [231], there are three key agreement categories: static-static, static-ephemeral and ephemeral-ephemeral. On static-static, there are no ephemeral keys on usage; on static-ephemeral, it only generates one ephemeral key pair for one of the parties; and finally, on ephemeral-ephemeral, each party generates an ephemeral key.

In general, a static key stays the same over a long period. However, an ephemeral key has a very short lifetime and is re-created for each session. The scheme ”static-static” does not provide forward secrecy, which means that if an adversary finds either one of the private keys, then the shared secret can be calculated (using the other party’s public key), and all security is lost. The ephemeral-ephemeral scheme provides forward secrecy, which means that past sessions are still secure even if an attacker finds one or both private keys, as this scheme generates a new key-pair for each key agreement. It is not accurate stating the ECDH does not provide authentication while using ephemeral keys, although it is required to authenticate the public key’s exchange. However, the protocol can use ECDSA, which is the elliptic curve form of Digital Signature Algorithm (DSA), to solve the lack of authentication. DSA authenticates digital content because a valid digital signature gives a recipient reason to believe that the message was created by a known sender, such that the sender cannot deny having sent the message (authentication and non-repudiation) and that the message is the same - without changes (integrity). This authentication includes the key agreement parameters used to derive the master secret and includes the session key's correctness. In brief, it can authenticate the handshake of the TLS protocol so that it can provide authentication for ECDH [232].

Finally, the scheme ephemeral-static does not provide forward secrecy because if an adversary finds the static private key, then the shared secret can still be calculated. However, we choose this scheme with the static key pair stored on the manager because there is no hardware token with support to ephemeral keys, to the best of our knowledge. As the secure token stores the private key offline, an attacker needs to have physical access to the physical token and know the Personal Identification Number (PIN) of the secure token to access the private key.

Middleware's Authentication Process

The authentication process is depicted in Figure 4.1 and shows its four distinct phases.

After the manager sets up the OTP Server, the manager is ready to receive new authentications from IoT devices. A new device can send its ephemeral public key to the manager to initiate the conversation. As the key is ephemeral, the device needs to authenticate itself with the manager. To achieve this goal, it uses the manager's public key (stored on the secure token) and its ephemeral private key (correspondent to the ephemeral public key sent on the first step) to derive the ECDH shared key. Then, the device sends to the manager a Certificate Signing Request (CSR) generated with the ECDSA private key and an OTP generated with the secure token, both encrypted with the derived shared key. The server also derives an ECDH shared key with its private key and the device's ephemeral public key received on the previous step. The manager can then decrypt the device's communication that contains the OTP and CSR. If the OTP is considered valid by the OTP Server, he knows that the client is in the physical presence of the physical token. If this is the case, the manager signs the CSR, generating a signed x509 certificate, and sends it back to the client. It attests that the new device is now on the device's trusted pool. These certificates establish trust among the client devices (without the intervention of the manager device).

Note that the secure token has a PIN to protect the signing action to ensure that no one, except the owner, uses that token to sign.

After this process, the shared key must be discarded (deleted) from the devices.

4.1.3 Discovery Process

After a successful authentication, the device now belongs to a new domain. We need a way to ensure that other services or entities can discover the new device inside the network.

We chose to use *Zeroconf* [233] to support the entire discovery process within our solution. It is a commonly used technology with a wide commercial adoption by companies, such as Apple, which uses *mDNS* to locate any connected speakers, Apple TV, and others.

So, on this configuration, the domain of devices is changed from *IoTDevice_1* (for example) to *IoT-Device_1.pool1* (Example on the Figure 4.1). When advertised on the network, the additional nodes can recognize them from the same domain. In the beginning, only the manager has his hostname along with the domain (pool1, chosen for this paper). However, all the trusted authentication devices must have the same domain when there are more authenticated devices. At this stage, devices can make a peer discovery to find devices with the same domain.

4.1.4 Decentralized Secure End-to-End Communications

After the discovery process, devices need to authenticate among themselves without the intervention from the manager. For mutual trust, both devices must exchange the manager's signed certificates. When the device *IoTDevice_1.pool1* wants to communicate with others, such as *IoTDevice_2.pool1*, they need to exchange their certificates to prove to each other that they are trusted.

As all the clients keep the manager's public keys, their certificates' signatures can be verified mutually. Then, both clients exchange ephemeral keys, signed with ECDSA. With access to the certificate signed by the manager, they can extract the public key and verify the signature of the ephemeral public keys to prove that it is the same person who has them (and, therefore, is authenticated). After exchanging their public keys, each customer can derive an ECDH shared secret to communicate (with their private key and the other's public key).

The security of the transmission of the *.crt* is implicit given from the possession of the private key that only the owner has access, so the authentication is guaranteed (even if someone eavesdrop on the channel and stole the *.crt*, does not have the private key associated to it).

4.1.5 Merge Two Trusted Devices Pools

Both identity and authenticate systems must be flexible and highly scalable enough to handle billions of device infrastructures in multiple environments such as smart homes and smart cities in general. These systems must support different environments, given the heterogeneity of applicability in IoT scenarios.

For greater scalability, there should be a proper way to integrate different device pools to make the system more practical as it would not be feasible to re-provision devices already provisioned with another manager so that devices from different pools can communicate with each other.

The system addresses scalability issues by replicating the traditional mechanisms of having multiple CAs supported by a client. It is essential to ensure two requirements to deploy this in a real-world configuration: use a secure token authentication scheme to enable enrollment and trust between different managers; and information dissemination on new pools among all new devices.

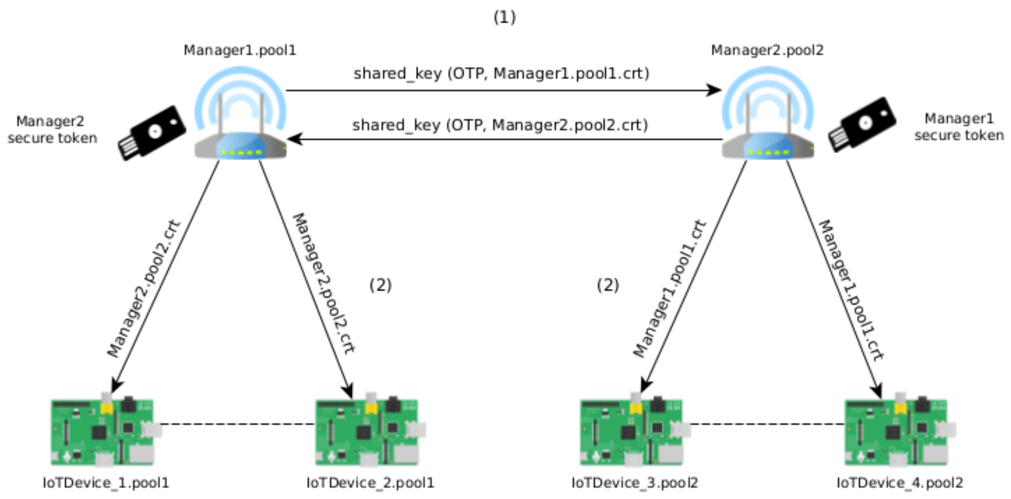

Figure 4.2: Trust between two device pools

It is necessary to have authentication between managers to allow two pools to connect (Figure 4.2 in 1), and the process is the same as explained in Section 4.1.2. After both managers perform mutual authentication, the next step is to spread the information across devices among different pools. To do this, the manager must send the signed and encrypted information to the device, allowing other devices to read the information and verify the manager's signature. Figure 4.2 represents the agreement between both managers and the corresponding spread of information from managers to their peers when they begin to trust each other and announce on the network that others should move to include these new trusted peers in their trusted network.

Now, when a peer from the new trusted pool communicates with them, they know they can trust them.

4.1.6 Certificates management and Information Dissemination

Certificate revocation is a process that needs to be quickly propagated to all devices. For example, revoking the certificate when there is a private key leakage is essential to guarantee the system's security. In this case, this process is essential to protect it from attackers who can gain access by identity theft.

We can revoke these types of certificates by removing certified copies from an allowed list. When there is an update on revoked certificates or an existing pool adds a new device, all other nodes belonging to the network (or the set of authenticated devices) must receive the updated information.

Given the high dynamism of almost any IoT scenario with highly intermittent connectivity and constant changes in density, data dissemination becomes a challenging service. The dissemination of information is not directly addressed in this thesis as it is related only to the devices' provisioning.

However, we need to describe this issue because we have the revocation and addition of new devices on this thesis's goal. We will assume a mobile device that disseminates information or fixed points to disseminate the information at the end of the day (or at a defined fixed time).

4.2 Evaluation

To analyze the impact of this proposed solution, both in terms of latency and resource consumption, this section analyses the performance of device provisioning by describing the entire configuration phase, implementation details, and an evaluation with execution times and energy consumption.

4.2.1 Setup

For implementing a PoC of our solution, we use three RP3 Model B+ and a Yubikey NEO. The devices connect to a UniFi AC Pro AP to mimic a common WiFi deployment. The Yubikey NEO represents the secure token. One RP3 works as the manager (that acts as a router) and the other two as clients, with all the RP3 having the Raspbian operating system.

As the OTP Server can represent a SPOF, we created it as a service that can be turned off, when not used, to avoid the possibility of centralized attacks.

We assume that the RP3 connect by power and do not use extra batteries, such as power banks. For measuring the energy consumption, we use a direct plug-in power meter from *efergy*.

4.2.2 Implementation

In this section, we describe the implementation details regarding the Yubikey cryptographic algorithms configuration, the local certificate authority, and finally, the OTP Server.

Yubikey Cryptographic Algorithms Configuration

For authentication, we choose to use a Yubikey that allows to generate OTP and supports Personal Identity Verification (PIV) [234] card interface. PIV enables RSA or ECC sign/decrypt operations using a private key stored on a secure token, such as smart cards, through the *PKCS#11* engine. *PKCS#11* bridges the gap between OpenSSL and Yubikey. A PIV-enabled YubiKey contains different slots capable of holding an X.509 certificate and the accompanying private key.

For the authentication process, we use Python 3.2, OpenSSL, the OpenSSL *PKCS#11* engine from *OpenSC*, the *p11tool* from *GnuTLS*, and the Yubico PIV tool for interacting with the PIV application on a YubiKey. With these tools, we can build a CA generated inside the Yubikey through the *PKCS#11* support.

This work uses a Yubikey for the OTP generation and store the private keys for signing and encryption processes. We focus on two Yubikeys' slots, mainly: 9c (for digital signatures) and 9d (key management). In a nutshell, we will use slot 9c for certificate signing purposes, and we are going to use slot 9d for encryption for confidentiality purposes, therefore, to decrypt content using the private key stored on Yubikey. Both slots require a PIN to perform operations with the private key. On the 9c slot, we create a 256-bit ECC key pair to use a 256-bit ECDSA. On the 9d slot, we also create a 256-bit ECC key pair for encryption and decryption. If the slot holds the Elliptic Curve (EC) key, we will perform ECDH and return the shared secret.

Local Certificate Authority

For setting up the certificate authority, it is necessary to have some files located in a folder to store the certificates generated, and there are some main files and folders for that. The certificates are stored in a *certs/* directory and are listed in the *index.txt* file. The first field describes the certificate status, i.e., V for valid certificates and R for revoked certificates, the second is the issuing date, the third has the certificate serial, and finally, the last one contains the certificate name (Organization and Common Name). The manager device stores all this information.

OTP Server

The OTP Server is implemented using *privacyIDEA* [235] that is a modular authentication system. It is possible to use *privacyIDEA* to enroll the secure token by using a test account and the *"Enroll Token"* in the *"Yubico AES mode: One Time Passwords with Yubikey"* option. This system allows the administrator to revoke or disable registered tokens. After the enrollment of the manager's Yubikey, it is possible to authenticate against the manager with the Yubikey.

Devices attempting to authenticate run a standby service to read a public key from the *stdin* of the Yubikey (to encrypt the communications) and to perform the communication with the manager.

4.2.3 Results

The exchange of data between the manager and the device is the main process during device provisioning (Figure 4.1).

This section presents different results regarding the sending and encryption execution time of the OTP verification and public key. We do not measure the time required to create the sockets, but rather the time of sending the encrypted and decrypted data and, respectively, the process of encryption and decryption. Also, by sending an HTTP request to *privacyIDEA*, we measure the execution time for validation of OTP. We use the *httpie* command to call the */validate/check* endpoint.

We collected ten-time samples from the provisioning phase for the manager and the device.

From the manager interactions, we have a mean runtime $615.1ms \pm 9.01$, while in the client interactions, we have a mean runtime $522.7ms \pm 56.1$. The overall provisioning has an average runtime of $1137.8ms \pm 65.11 + \delta$, where δ is the time required to insert the PIN when performing the cryptographic operations.

We calculate the verification's runtime with *time.time()* on the authentication's Python implementation, which returns the system date and time.

In terms of energy consumed, the RP3 spends $2.2Wh$ only on without running services. When we did encryption and decryption, the energy spent was not significant, having just varied $0.4Wh$ up and down.

The energy consumed by the device manager when the *privacyIDEA* service is running and on the OTP validation process was $2.2Wh$, not varying more than $0.1Wh$ up.

4.3 Use Case Discussion

Smart Homes
This concept of authentication works in smart homes, in which there is a manager (which can be represented by the router) that authenticates home devices. These devices, once authenticated, can communicate with each other. The domain of these devices is associated with the pool to which they belong.

In a healthcare context, we can change the manager to be, for example, a smartphone that belongs to the manager of each department, and each manager authenticates the devices belonging to each department. Two different departments can communicate with each other if their managers authenticate and add each other's pools as mutual trust.

Smart Cities
This work applies to the sensors scattered around the city for authentication between them, making islands of devices connected.

We can imagine a certification chain where the city is the root CA, and each manager can authenticate with the city manager, becoming a sub-CA, which means that when devices authenticate with the manager, they are also automatically authenticated with the city. This way, we managed to have different independent pools, in which these devices only rely on devices that have certificates in which the city is part of the certification chain.

Before connecting to the city, the manager is a root CA for his devices, but he becomes an intermediate CA when he connects to the city. With this, the user's devices' certificates will have the city in the certification chain, which means that the devices can communicate with each other.

With the use of Yubikey, we guarantee that the city cannot accept provisions for non-manager devices.

What if there is a case of certificate revocation and adding a new device while on the move? As we said in subsection 4.1.6, one way of disseminating information is through a mobile device. In a smart city scenario, the dissemination of information, in addition to being done only at certain times of the day, can be done, for example, by a taxi or city bus (to try to be more efficient and faster to propagate).

4.4 Security Analysis

This section is defined as a threat and an attack model with a security analysis of the proposal, exploring the vulnerabilities from the defender's and attacker's point of view.

4.4.1 Threat Model

This section identifies system (assets) and potential threats against the proposed system from the defender's perspective.

Physical devices
The only SPOF in the manager is the OTP Server. However, this service can be turned off whenever it is not necessary to provision new devices.

If an attacker stoles the security token, it is possible to revoke the OTP Server's secure token. Also, it is possible to set an expiration time.

Cloning a device is a problem for current IoT devices. When there is physical access to any device, security can be compromised. Current research challenges focus on deploying trust zones, such as Intel Software Guard Extensions (SGX) [236], where parts of the code can safely compute secrets and store authentication keys and device identity. Future work will focus on mitigate and ensure device integrity.

On the other hand, using a secure token helps protect against hacking as physical access to the secure token is required to generate OTP.

Configurations for mitigation of attacks
There is an option within *privacyIDEA*, the Maximum Fail Counter, to avoid brute force attacks when an authentication request occurs. If the fail-counter exceeds this number, Yubikey's user cannot use the token unless it resets the fail-counter. This system has a fail counter of 5 attempts, and this option must be manually reset to prevent this type of Brute Force attack.

The private key never leaves the Yubikey and can only be used by the owner that knows the PIN. Even if the Yubikey is lost or stolen, the PIN has three attempts, and, if failed, it requests Personal Unblocking Key (PUK) that also has three attempts. If all fails, the user must reset the Yubikey, which erases all the content from the Yubikey. For this reason, owners' should also make a backup of the information contained on the Yubikey.

4.4.2 Attack scenarios

This section identifies attack modeling from an attacker's perspective to analyze the possible vulnerabilities on this system.

The secure token generates OTPs and stores the manager's public key to transmit to the trusted devices, which prevents the manager from being impersonated even if the attacker could change his name. For this reason, the tag impersonation attacks are safe because an attacker must have their data (authenticated OTP and its public key) in the secure physical token, which is impractical.

The system is also safe from replay attacks because the secure token uses a set of volatile and non-volatile counters that ensures that an OTP can no longer be used after validating once [237].

For man-in-the-middle attacks, there are two attack vectors: unknown peers and authentication peers. This approach uses a secure token with the receiver's public key, so it is transmitted only over Universal Serial Bus (USB), mitigating the unknown attacker's access to the public key and the OTP that authenticates with the OTP Server. For authenticated peers, as already described in the replay attack, the problem is solved using an OTP that can no longer be used after validating once [237].

4.5 Summary

This work presents a design and implementation of a new approach to IoT device provisioning, giving an identity to a *thing* and authentication between devices, eliminating the risk of impersonating attacks.

Often, the reluctance to use these types of solutions is based on smart cards, which require external readers on the computer. However, this work uses a secure token that only requires a USB port, and given the study by authors *Kiran Jot Singh and Divneet Singh Kapoor* [238], there are USB ports on most devices used in IoT. However, it is possible to adapt this system to other technologies, such as RFID.

This work proves the solution's feasibility by presenting a wide range of options to be deployed in real-world scenarios.

This solution loses in the provisioning time with an average of $1138ms \pm 65.1$ against the pTASC runtime, which is $1050ms \pm 58.2$. However, the difference is not significant, and this solution guarantees the authentication of devices, providing more scalability than pTASC. Besides, with KEAV and pTASC solutions, we could not guarantee an identity to the device, so that it was easy to manage revocation and dissemination of information between authenticated and trusted devices.

With this solution, we have several benefits: device identity, scalability, offline cryptographic assets, revocation and dissemination, and resistance to MiTM. It is possible to have device islands, which in addition to authenticating multiple devices, can authenticate with other device islands (for example, different entities or departments), allowing to scale the solution to provision and authenticate multiple devices.

Chapter 5

Empowering Users Through a Privacy Middleware Watchdog

”Privacy is not something that I am merely entitled to, it is an absolute prerequisite.”

— Marlon Brando

The amount of information that a IoT device can collect is substantial. Webcams can see everything, smart TVs and personal assistants can hear everything, and smart cars can give clues as to whether a person is at home or not. The amount of data that an IoT device can send back to its manufacturers and how they are stored depends solely on them. Most of the time, users are unaware that this information is being sent and shared with external sources. These data can still be intercepted or forwarded to a malicious server if not properly protected. In addition to sound and image, depending on the device, data sent to external sources may include sensitive information, such as the IP addresses, other devices connected to the network, and location. Some manufacturers may collect confidential users’ information and gather patterns about their lives (whether they are at home, the content of their conversations, and others).

Several articles explore the use of voice assistants and feature emerging privacy issues from them [239]. In most cases, users cannot control their data, nor are they aware of data sharing to external entities. *Lau et al.* [240] claim that, despite this and even with the possibility of devices’ privacy settings, users do not have enough knowledge to access and edit these privacy settings and often prefer to turn off the voice when they are talking about more private things in the same room, because they do not trust that the assistant is not even listening.

There are numerous devices from brands, such as the Amazon Echo [241], Google Home [242], Apple HomePod [243] or Amazon Echo Dot [244] smart speakers. Millions of smart speakers (from Google in this case) have been sold [245], and worldwide spending on these wireless smart speaker devices reached the $2 billion on 2020 [246]. A vast portion of the population is using these types of devices in their homes, but what are the privacy implications to them? Of course, we are not saying that all these devices disregard privacy policies, but the truth is that with a microphone on, there is always an intrinsic fear.

Josephine Lau et al. [240] shows that many users who have voice assistants say they do not care about privacy, but when faced with the possibility of hearing everything they are saying, they prefer to hang up and not use it. Many users who do not have a voice assistant guarantee that they do not trust these devices regarding privacy.

The biggest overall concern is the lack of information about the user's privacy and that there are no appropriate settings for, for example, turning off the microphone without disconnecting the entire device from the Internet. There is no transparency of the communications made by the device to the Internet, and, therefore, there is no information about information shared. A subset of the users inquired in this study referred to the possibility of having detailed information about when and with whom the data is being shared [247] and being able to do the local processing without the data being sent for public clouds.

Focusing on smart-home contexts, how can we build a solution that allows users to control their data and deal with privacy? This is one of this thesis's goals, which aims to provide users with ways to control it.

This chapter proposes a middleware layer that allows users to control the data generated by their IoT devices. Depending on the manufacturer's firmware, users can store specific data offline and control data sharing while preserving their privacy. The middleware should also behave as a data broker, implementing semi-autonomous and autonomous strategies to control IoT device's traffic, probably considering factors such as the data owner's preferences.

We perform this middleware integration on a testbed with a RP3, and a more realistic scenario with a Google Home and a Phillips Hue for demonstration purposes. This implementation uses the Google Smart Home actions to allow users to control their devices through the Google Home app and voice commands with the assistant. We used this implementation to deploy a real-world usage scenario with the Google Home device and Google Assistant API with some sensors on the same home network. We present the middleware structure integrated with this application scenario, implementing all features related to the security of communications, namely at the authentication and identity of each device (included in Chapter 4) to guarantee secure and private communications between them.

5.1 Conceptual Design Overview

Secure and privacy-oriented data sharing remains an unsolved problem in IoT, especially for users' data. Users are generally unaware of how their data is being handled in the IoT environment, as they assume that the manufacturer implements the appropriate privacy and security mechanisms. However, this is not the case, mainly from the examples described in the introduction section 1.

In our approach, users can decide the data and traffic exchanged according to their preferences. Users have the option to block all traffic by default and to make exceptions for some specific domains. Therefore, users can block marketing/advertising sites and only communicate with the manufacturer's domains. If users choose to block all communications from their devices to the Internet, some of the features may stop working, as some of these devices will not work in offline mode. In this case, users will have to choose between usability and privacy.

Existing routers have a set of rules to block specific traffic. However, the difference to the middleware data sharing control is that there is no easy way for users to consult privacy settings that show connections made by default by the device (configured by the manufacturer) and a way to block these connections ”in real-time”.

Along with this network traffic monitoring and, depending on the manufacturer's device firmware, users can store data offline on their home router for future reference. As users can connect to multiple routers (at home or work, for example), they can choose different permissions for their data depending on the device's context, managing the data life cycle. Users can also monetize data by selling it to external entities.

5.2 Architecture

From an architectural point of view, the system includes IoT devices, the owner's smartphone that acts as *Permissions Controller*, and a router that acts as a manager and controls data sharing for external entities.

Figure 5.1 shows the different components of each device, as well as the interconnections between them and the Internet.

There are two types of IoT devices: white-box and black-box devices. White-box devices represent devices on which it is possible to install software, modify and access the firmware. On the other hand, black-box devices represent devices on which developers can not control the stack and do not have access to the software/firmware, so it is impossible to install applications with the level of granularity necessary to install and run the middleware.

5.2.1 White-Box Devices

This type of device has two main components: a *middleware* and a *Sensors API*.

Sensors API has the necessary interface for IoT devices. Some manufacturers allow developers to interact with device data in their applications, products, and services. In such cases, the *Middleware* component integrates with the *Sensors API* to get access to the data. Then, the *Database* receives

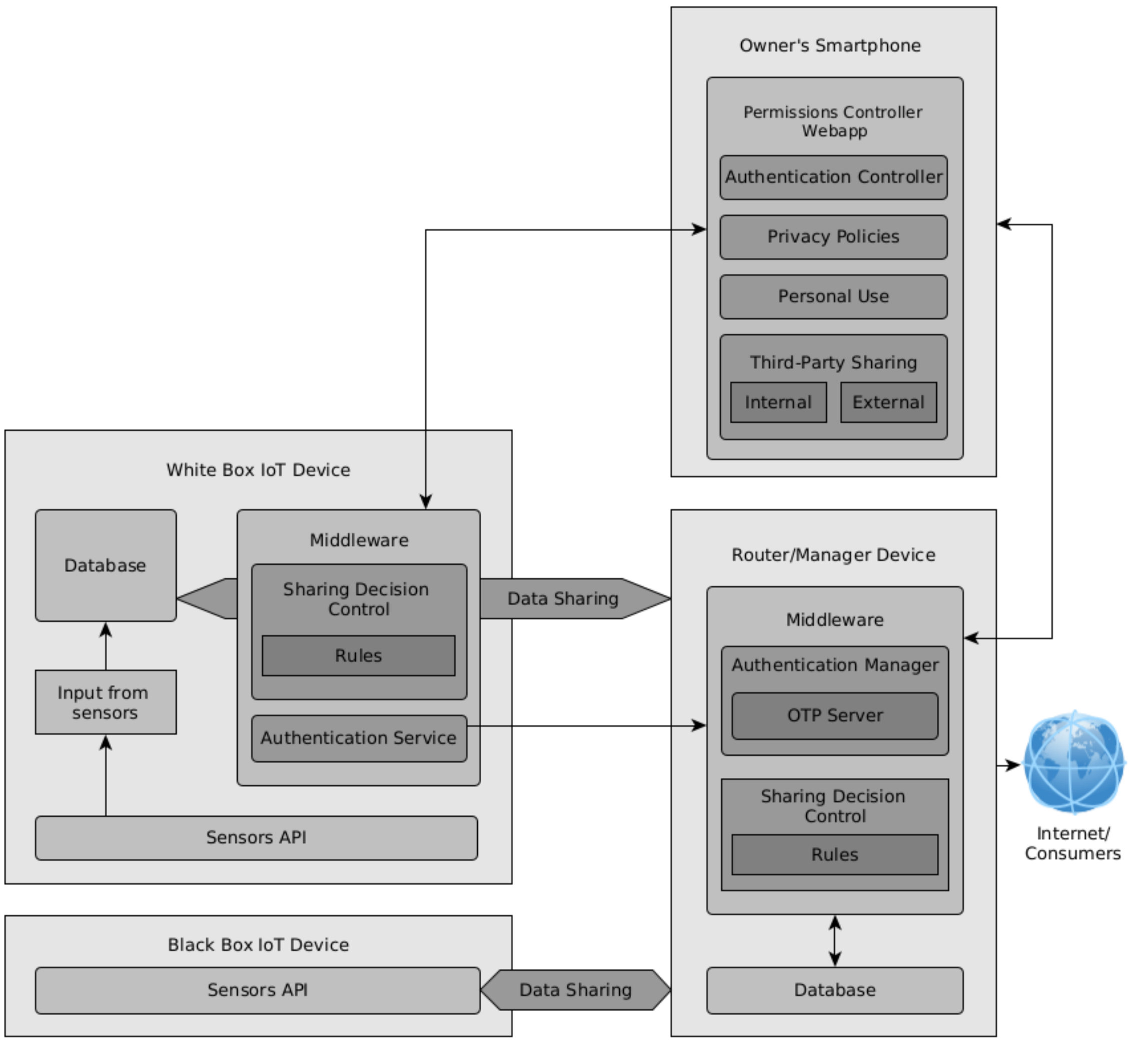

Figure 5.1: Data Sharing Middleware

the input from the API and can store the data locally (this component stores data for seven days maximum by default, but users can modify this configuration).

Sharing Decision Control consists of a firewall-based solution to control data sharing. The component uses context-aware access control and owner rules provided by the *Rules* component to control data sharing. The *Sharing Decision Control* must be deployed locally on the device to provide the owner with the security of their data under their control.

5.2.2 Black-Box Devices

When the manufacturer's firmware is closed/proprietary, it usually uses encrypted channels to support all communications, creating an opaque layer where it is impossible to distinguish between the transferred data. Thus, it is impossible to implement the middleware on the device, so the router must perform all the traffic control (the router also has the middleware).

5.2.3 Permissions Controller

Our choice to use smartphones as a configuration controller to define rules and permissions aims to promote usability, as screens or keyboards are generally not used/available in IoT scenarios.

For black-box devices, the smartphone sends rules to the router, as it is impossible to control data sharing locally. For this reason, the only option available is third-party sharing for external sharing. *External Sharing* is related to the traffic that the device generates by default, which may include communications with the manufacturer and entities unrelated to the manufacturer. In the case of black-box devices, the traffic is encrypted, and most of the times, devices communicate directly with an external server, and so, transferred data are not distinguished, but with this option, users can select the entities they want to share with (for example, *Google* is allowed, but *Facebook Ads* is not allowed).

Users can choose between two modes: *primary* or *advanced mode*. The *primary mode* has traffic data aggregated by the entity. In a use case based on *Google Assistant*, device connections include multiple subdomains of *googleapis.com*, but only the *Google* entity appears to users in *primary mode*. In this way, all connections to or from *Google* are blocked or allowed, according to the users' preferences. This mode is more accessible for *non-technical* users to control the entities with which devices communicate. Thus, users can also know which connections are related to the device manufacturer.

In *advanced mode*, all traffic is shown by domain. According to the same use case, instead of showing only the *Google* entity with aggregated traffic, it shows detailed traffic in real-time (*2020-09-30 11:13:09 - www.google.com - raspberrypi-home.lan; 2020-09-30 11:13:16 - play.googleapis.com - raspberrypi-home.lan; 2020-09-30 11:13:16 - clients4.google.com - raspberrypi-home.lan*), and users can block or allow specific connections.

As mentioned in Section 5.1, devices that do not work offline and depend on communication with the manufacturer may not work after some traffic block. If only a few features stop working, users can choose between usability and privacy. In summary, in this component, the smartphone can display all traffic in real-time, and it is possible to choose the entities to share the data and specify the expiry time of these permissions.

For white-box devices, each owner has four main permissions to control the data sharing: *Personal Use*, *Third-party sharing*, *Sell Data* and *Privacy Policies*. In this case, both the IoT devices and the router implement all the smartphone rules.

On the web app, users can access these controls on any network device with display capabilities.

Personal Use

Personal Use has two options: *yes* or *no*. The *Personal Use* options allow users to store their data (produced by the IoT devices) on the router's database. When users set the permission to *yes*, the router's database can store the data produced by the device. This option allows owners to control their data locally without an Internet connection, creating their API and services.

The Web App allows users to query this data to be consulted for further analysis. This option is only available for White-Box IoT devices because we cannot get data encrypted in both ends on the Black-Box IoT devices.

Third-party Sharing

Third-party Sharing has two different components: *Internal Sharing* and *External Sharing*.

Internal Sharing represents the data transmission between the IoT device and the router to which it is connected. Owners can set different permissions according to the IoT device. For example, if the device is in the *home* context, owners may want to store all data on their router but deny this permission in other contexts blocking the transmission of some confidential data. The owner's smartphone shows all sensors available for data sharing and allows users to set permissions for their data according to their preferences, enabling selective disclosure of information to ensure anonymity according to their preferences. Such selective disclosure should consider the appropriate, relevant, and limited data to what is necessary for the purposes by providing minimal disclosure of personally identifiable information or other sensitive data types. This concept can be called data minimization by the new GDPR [248]. This option of Internal Sharing is only available for White-Box IoT devices.

External Sharing is explained in Section 5.2.3 and is also available for these white-box devices. In summary, *External Sharing* is related to the traffic that the device generates by default (communications configured by the manufacturer) so that users can manage these communication permissions. With this option, users can manage (allow or block) those connections made during normal device operation but cannot choose other entities to share data.

Sell Data

Unlike *External Sharing*, which controls only standard device communications, the sell data option allows users to sell specific data to interested third parties. The sensors can gather additional data that may be of interest to marketers to reinforce marketing strategies in a region, creating accurate and personalized ads contextualized to a person or region's interests. Our structure allows users to choose a set of data collected by the IoT sensors and stored in the router's database to be shared with external entities (also selected by the data owner) in exchange for monetary compensation, offering users the possibility to monetize their data. For example, users can choose to sell data about temperature but not about lighting and air conditioning because they know that a machine learning algorithm can combine data to determine the presence in a given household. Note that this sell data option is valid for white-box devices or devices with data APIs.

Privacy Policies

Each home or workplace has an owner configured on boot time as admin. The owner then has a set of configurations to control rules and permissions on the webserver application. An owner can set other users' permissions connected on the same network.

Users can have different permissions according to their hierarchy. First, users can authenticate both as household or guest. Guest users are the less privileged user because it can only consult the devices' privacy policies. For ease of use, it is possible to quickly set up a guest, as he can read a QR-code (which can be on the person's home wall), and these users can see information about the IoT devices on the house (name, type) and the data they are collecting. The information about the privacy policies is inserted manually by the owner when it sets up the house, and it includes an opting in or out for data collection, to the guest choose if they agree or not with the data collected.

This platform also guarantees the privacy of different household members, allowing personal privacy settings for each member, which must be preserved by the system, and provide settings that prevent private information leaks for non-household members, e.g. guests or neighbors.

New users can be home residents that the owner must approve. The owner can delegate or share administration capabilities associated with delegation of permissions or others to some *responsible* users by setting them with according permissions. Each home can have *several responsible* users that can manage the administration settings and data privacy settings of that home devices. These *responsible* users cannot modify the permissions set by the guest clients or owners.

By default, all the users can only define policies related to their data (e.g., users may not want to have their voice recorded by devices at the network owners' house.).

The network owner or authenticated users can set their data privacy policies for their devices. Alongside with the guest settings, it is essential to have more automation in some of these processes. Guest users can then deny microphone permissions for other IoT devices. For example, guests can opt-out of this permission, and the house's middleware blocks all communications from the microphone (if it is not possible to specify individual permissions for each resource, all communications from selected smart home devices to the Internet are blocked.).

If multiple people are on the same network and have different permissions, the deny permission will always have a higher priority order.

5.2.4 Router

The router has two different components: a *Sharing Decision Control* and a *Database*. The *Sharing Decision Control* component acts as a firewall between external entities on the Internet and local devices. This component receives the rules provided by the mobile application and controls data sharing.

When the device's *Personal Use* and the data's permission on the *Internal Sharing* component have the answer "*yes*", this data can be transmitted to the *database*, allowing data owners to query them offline. In the router, it is possible to connect the mobile application and the *database*, allowing users to query their data on the smartphone. For devices that continue to operate offline without connections to the manufacturer, this can be a way for users to view their data without interacting with the manufacturer's online application that stores the data in their proprietary database. In addition to this, the router also has a web interface that shows the data available for sale.

We choose to place data sharing control on both devices (IoT devices and router) to ensure that users have their data under control, independently of device that are they are interacting. As we have black-box devices where we do not control both endpoints of the encrypted data channel, we need to ensure that users can control their data. The only way to do this, is to have the middleware on the router so that users can allow or deny communications. Besides, this control allows router owners to have their internal permission policies. For example, when a new device connects to a router and does not block any traffic, the router owner can have internal default permissions to override this behavior and block such traffic.

5.3 Evaluation

In this section, we present the setup and details about the implementation of the prototype and the results related to energy consumption.

5.3.1 Setup

In order to provide a PoC, we build two separate setups. The first setup has four RP3, one Yubikey NEO, and a USB pen drive. The four RP3 represent two IoT client devices with four sensor modules: Temperature and Humidity (DHT-11), Luminosity (KY-018), LED (KY-016), and Heart-Rate (SEN-11574); one that represents a router and the last that represent the manager. All RP3 have the Raspbian operating system. The second setup is equal but includes a Google Home and a Phillips Hue light.

On the RP3 that represents the router, we build an access point. In this operation mode, the WiFi serves as a connection point for other devices (clients/stations), similarly to a conventional home router's operation. The most common purpose for WiFi to operate as an Access Point is to connect to other WiFi devices, share connectivity to the Internet and create a dedicated network for some configuration purposes (as in some home routers, for example). This device serves as a network scanner to keep track of the hosts connecting to our local network and to be able to make network filters, allowing the control of incoming and outgoing traffic.

The router also acts as the manager device with the OTP Server. This way, when the owner is not provisioning new devices, the OTP Server should be turned off to prevent the possibility of an attacker stole the private key.

To represent the secure token referred to in Section 4, we used a Yubikey NEO. We use power to connect the RP3 instead of extra batteries (power banks). To measure energy consumption, we used a direct plug-in energy meter [249].

5.3.2 Implementation

This section presents details of the implementation of the proposed system. We describe the technical details about the sharing decision control component, including internal sharing with context-aware and authentication mechanisms and integrating the sell data component and marketplaces.

Sharing Decision Control

Our proposed solution focus on DNS filtering by implementing a system based on *PiHole*. It aims to block ads and tracking mechanisms on the Internet by resolving domain names to IP addresses, and if these domains belong to a blacklist of domains used for advertising or tracking will be barred. In this way, all devices work with *PiHole*, which controls each device individually, allowing them to have different permissions and filters for each.

One advantage is that we can select different permissions according to the Media Access Control (MAC) address (each device). When dealing with White-Box devices, it is possible to install a middleware in each device, and this way, the control is inside the device and for each device. However, on a solution with black-box devices, it is necessary to make all the control on the router. This way, we need to filter specific permissions for each device, with their MAC addresses.

PiHole does not have a native DoH capability. However, it is possible to configure a DoH proxy, such as *clourflared* or *DNSCrypt* and point that at a DoH resolver. The *PiHole* would then query the proxy, which would query the DoH upstream resolver. With this properly configured, it is possible to block some traffic according to the users' rules.

We will not deal with TLS1.3 in this thesis, but we decided to adopt a solution that already has a basis for the adaptation.

Context-aware Internal Sharing

Context consists of information that can characterize a physical or situational environment (set of circumstances). It is also considered any information that can characterize an entity (for example, a person, place, or computing device) [250]. IoT predicts an era when billions of sensors connect to the Internet, which means that it is not possible to process all the data collected by these sensors. However, there must be a way to decide what data to process (context awareness) and store information of context linked to the sensor data to make interpretations of each situation more easily.

Many sensors increase every day, which generates a large amount of data. These data are often of no value unless they are analyzed, interpreted, and understood. Understanding the context facilitates communication between machines, as it is a central element of the IoT vision [251].

Context-aware systems can dynamically and automatically adapt to changes in the environment and users' current needs without requiring their attention. There are several definitions of context, such as ambiance, attitude, circumstance, dependence, environment, location, occasion, perspective, phase, place, position, posture, situation, status, standing, surroundings, and terms. There is much research on context-aware computing over the years [251–253], especially related to self-learning techniques in IoT and decision making.

In this work, we introduce an approach with different identities and access control policies for a device. The initial goal is to understand the device's context and what types of context we want to address. As an example, users can set different permissions for their smartwatches, depending on the context (for example, location: at *home* or *work*). In this example, users can decide not to share their heart rate in the *work* context but do not mind sharing the number of steps per day; on the other hand, in the *home* context, users can decide to allow the storage of all data. In summary, context-based techniques are essential to automate some privacy settings in decision making.

However, there is a drawback to this approach, as it can be attractive for attackers, trying to set the user location to *home* when the user is actually at *work* or *mall* for opening the user's home window, for example [254]. To mitigate some of these risks associated only with the context, in addition to the authentication process, we use a set of sensors with some extra information, namely WiFi, Bluetooth, and GPS, instead of using only one sensor.

In addition to these conditions, it is essential to detect some possible abnormal patterns autonomously. For this, the middleware creates a database with a history of usage patterns on each device. Thus, if the middleware receives a request for permission to open a window due to the user's presence at home, at a time when the user is never at home according to historical usage patterns, the middleware should notify and request the user intervention manually to decide whether the window will open or not. In this example, this feature is vital to ensure that we are not facing a geolocation attack or other types of attacks based on changes to sensor data that influence context-based decisions. Therefore, if any of the predefined conditions fail (in terms of sensor data and usage patterns), the user will need to intervene and grant specific permissions for those specific cases manually.

Sell Data

This concept focuses on controlling data sharing on the user side, not necessarily on the market itself. The idea focuses on linking the middleware to an external third-party marketplace that does all sales management. For demonstration purposes, we integrate with the implementation of a market created by *Xiangchen Zhao et al.* [255]. With this implementation, user registration is manual, and users need to provide the data to the market for sale. For this, the middleware is ready and provides users with a well-defined API with the data they choose to sell.

5.3.3 Results

Regarding the performance and energy consumption results associated with the authentication process, we should refer to the Section 4.2.3.

In terms of energy consumed, the RP3 spends $2.2Wh$ without services running on the system. With the middleware running on devices and a database, the energy consumption remains almost the same. When the middleware performs encryption and decryption, the energy spent is not significantly different, varying only $0.4Wh$ up and down.

Regarding the network performance, with *PiHole*, we did not have a latency increase. As *PiHole* only handles DNS, it does not process every bit of data that users exchange on the Internet. The traffic still flows between the client and his router to the Internet, and nothing goes through the *PiHole*. As DNS traffic has very low bandwidth (a few hundred bytes per request), the *RP3* will be able to handle them easily, whether on the wired or wireless interface.

5.4 Testbed Evaluation

The study of scenarios and the integration of implementations allows us to have the necessary experience to identify the common denominator at various abstraction levels, aiming to develop a middleware that factors these standard functionalities and provides a productive tool.

In this section, we present an overview of the Google Smart Home APIs.

Google smart home lets users control their connected devices through the Google Home app and the *Google Assistant*, which is available for smart devices, like smart speakers, phones, cars, TVs, headphones, watches, and more. The *Google Assistant* SDK allows developers to incorporate *Google Assistant*'s features into their products, such as hot word detection, voice control, natural language understanding, and Google's smarts.

More recently, Google also created the Local Home SDK that enables the fulfillment of smart home actions directly on Google Home or Google Nest devices and enables communication with smart devices over the local network to fulfill user requests. This creation was due to latency and reliability when engaging the user's action. Users can write and deploy a local fulfillment app in Typescript or JavaScript to identify and execute commands on any Google Home or Google Nest smart displays, deleting the developer cloud's dependency.

Google Home is a smart speaker with a microphone that has Google Assistant installed on it. It cannot work without Google Assistant. However, Google Assistant can work without Google Home. Google Assistant works both on Android phones and on RP3. Google Assistant capabilities depend on the devices where it is running.

Throughout this section, we also describe the integration between Google Home, the middleware, and IoT devices and the integration of the Google API on a RP3. These two testbeds allow us to show that it is possible to use the middleware either with black or white box devices, with different functionality. Also, we use these testbeds to validate the solution of the sharing decision control and the integration between components, namely the authentication itself for white-box and black-box devices.

We use a combination of tools for the network monitoring, namely the *tcpdump* and *moloch*. *tcpdump* is a tool that allows "sniffing" all traffic that passes through the data network. We use *Moloch*, a full open-source packet capturing, indexing, and database system, to analyze network packet capture files. It works as a viewer to parse and view the full packet capture done by *tcpdump*.

5.4.1 Google Home and Phillips Hue

This integration focus on exploring a Black Box device architecture because we are dealing with the Google Home that is closed/proprietary, so it uses encrypted channels to support all communications, creating an opaque layer where it is impossible to distinguish between the transferred data. Thus, it is impossible to implement the middleware on the device, so the router must perform all the traffic control (the router also has the middleware).

Middleware's primary goal is to protect the user's privacy in the context of IoT. This middleware works as an emulator of a device that controls the smart home devices of users. With this, it is not Google that has control over our device (that is, Google Home does not give the executions directly), but instead, the middleware receives the action and controls whether the action affects the IoT device (Figure 5.2). Basically, instead of having the users' IoT devices on the *Google Assistant* app (controlled by Google), the middleware is the device that emulates the IoT attached device, i.e., the middleware receives the order and gives it to the IoT device (users have the control of the devices locally through the middleware).

We create a complete integration to evaluate the practical impact of privacy. Also, we show how the authentication scheme integrates into this setup. This testbed was integrated from scratch using the respective API's available from the manufacturers. The devices were not rooted or any operation that might render them useless for general use.

5.4.1.1 Setup

We set up a Google Home to make a testbed with a real device, along with a Phillips Hue.

We use a RP3 that builds an access point for the router. In this operation mode, the WiFi serves as a connection point for other devices (clients/stations), similarly to a conventional home router's operation. The most common objectives of WiFi operating as an Access Point are to offer connectivity to other WiFi devices, share connectivity to the Internet and create an exclusive network for some configuration purposes (as in some home routers, for example). This device serves as a network

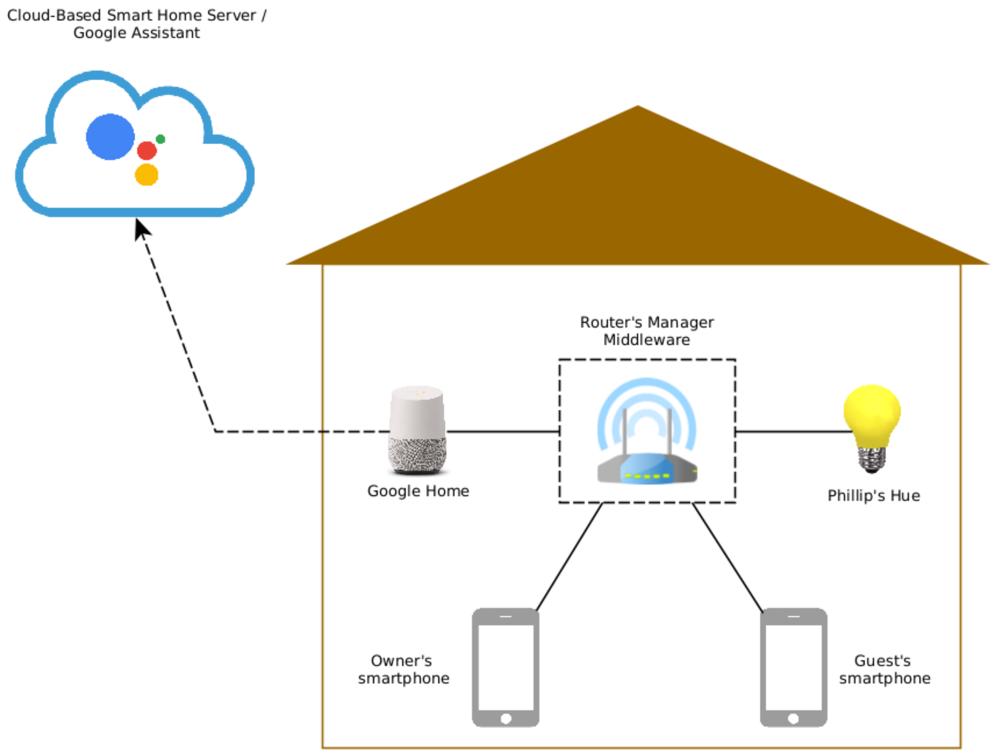
Cloud-Based Smart Home Server / Google Assistant
Router's Manager Middleware
Google Home
Phillip's Hue
Owner's smartphone
Guest's smartphone

Figure 5.2: Controlled Smart Home Environment

scanner to keep track of the hosts connecting to our local network and to be able to make network filters, allowing the control of incoming and outgoing traffic.

5.4.1.2 Network Traffic Monitor

We install the sharing decision control only on the RP3 that acts as an access point. It is not possible to install on each device because the Google Home and Phillips Hue devices are considered black-box devices, and it is not possible to install it there. For this reason, on the access point, we define specific rules for each device by MAC address.

We define a DNS regex $.*$ for block all the connections and we add to the white-list the regex: `(\.|^)google\.com$`

We tested the Google Home functionality together with the Phillips Hue, with that definition.

First, we said, ”Hey Google” and the Google Home replies, ”Sorry, I cannot reach the Internet right now!”. Then, we analyze the connections that the device made, and the device tried to connect with *www.gstatic.com* and *connectivitycheck.gstatic.com*. For this reason, we add two more regex: `connectivitycheck.gstatic.com` and `(\.|^)gstatic\.com$`.

After adding these rules, Google Home already answer to our requests. For that reason, we made several interactions:

- Turn the Phillips Hue lights on and off;
- Ask for information about the current and next day's temperature;
- Ask for the day's information news and try to turn on Spotify service.

These communications led to multiple connections that we can track on the *PiHole*. To turn the Phillips Hue on and off, Google Home sends the following DNS requests: *www.google.com*, *clients4.google.com* and *play.googleapis.com*. The light turned on and off, even with the *play.googleapis.com* blocked. Also, it adjusts the light brightness according to what the voice asked for.

Then, we tried to get information about the weather, and the only DNS request was *www.google.com*, and so, it works correctly. We tried two more requests, namely for Spotify and news.

For Spotify, we had to unblock some services related with **.scdn.co* and **.spotify.com*. The *googleapis* stayed blocked and works well, but we had to unblock the others completely. The DNS requests made were: *audio-fa.scdn.co*, *seektables.scdn.co*, *clients4.google.com*, *play.googleapis.com*, *gew-spclient.spotify.com*, *clients3.google.com* and *api-partner.spotify.com*.

Lastly, for the news request, the Google Home only presents the sources from the news, but does not gives the content of information, because all the external sources were blocked, namely: *radiodownloaddw-a.akamaihd.net*, *bbgvod-azure-us-east1-zenko.global.ssl.fastly.net*, *lh3.googleusercontent.com*, *pdl-iphone-cnbc-com.akamaized.net*, *lh4.googleusercontent.com*, *dts.podtrac.com* and *googleassist-cdn.prod.reuters.tv*.

5.4.1.3 Authentication

In this type of black-box device, where we cannot have local control of the device (due to the impossibility of modifying the firmware), it is necessary to protect the device's internal and external access in another way. Our proposal aims to segregate the devices in multiple Virtual Local Area Network (VLAN) other than the regular network, and only the middleware accesses these devices. In other words, the middleware controls the traffic that leaves and enters these devices.

This way, it acts as a local control to the devices, where only the traffic controlled by the user leaves and enters. Using the Google API, the middleware acts as the authenticated device and communicates with the Phillips Hue (that is inside the VLAN). Thus, users make all the control and have the last word to say about the access to their data.

5.4.2 Google Assistant on Raspberry PI

Another way to use the Google's virtual assistant, without the need to purchase Google Home, is through the Google Assistant. The most used form of this service is through smartphones. However, to simulate a Google Home, it is possible to transform a RP3 into a kind of Google Home ”hands-free”. More advanced users may prefer this type of solution because they can turn off the microphone, for example, in a physical way (making sure that the device not shares the sound that users make at home to the Internet). As software settings can easily be changed, only physical switches to turn on and off components can be wholly trusted to prevent data collection. For example, by removing a microphone from a device through the USB connection will act as a kill switch. The hardware kill switch is one option that devices must add, especially devices such as Google Home, Amazon Echo, and all those IoT devices with access to a microphone, camera, or another type of sensors that allow the collection of sensitive data. Of course, adding physical switches can impose extra costs and adds extra engineering, and, in some cases, the ability to cut the electrical circuits to components will significantly change the weight, size, and ergonomics of devices. However, it is a moral imperative to protect people's privacy and respect their digital rights.

For this reason, this may be a more privacy-friendly solution.

5.4.2.1 Setup

In this environment, we built a Google Assistant using the Google Assistant SDK to control by voice the LED connected to the RP3 GPIO pins. For this setup, we also plug a USB microphone and a set of speakers to the RP3.

5.4.2.2 Network Traffic Monitor

Google Assistant on RP3 is much more limited than Google Home. For example, Google Assistant not integrates Spotify, so to turn on Spotify, users must have to configure options such as the web browser, but the RP3 blocks the Digital Rights Management content. Therefore, it is not possible to start Spotify. It also does not support obtaining news through voice commands, such as ”Hey Google, tell me the news”, but it does support, for example, obtaining weather with ”Hey Google, tell me the weather for today”. For interconnection with other devices (black-box or white-box), it is possible to pair them by adding the middleware to the Google account (and, therefore, the actions are given to the other devices through the middleware). It would also be possible to place the devices directly in the Google Assistant application, but we have deployed with the middleware for privacy reasons.

We did the traffic analysis for the interactions of turning on the Phillips Hue lights, turning on a light on a RP3, and obtaining information about the weather on the current day.

With this, we obtained different DNS queries, all related to Google. That is, for the connection, as in Google Home, the query is made to *connectivitycheck.gstatic.com*, and the other queries, practically all linked to *googleapis*, were as follows: *oauth2.googleapis.com*, *embeddedassistant.googleapis.com*, *mtalk.google.com* and *android.googleapis.com*. Therefore, in this situation, the simple white-list with regex: `connectivitycheck.gstatic.com`, `(\.|^)gstatic\.com$`, `(\.|^)google\.com$` and `(\.|^)googleapis\.com$` allowed these interactions between home devices.

5.4.2.3 Authentication

The white-box devices, which can be sensors scattered around the home or a set of smart homes, can deploy the authentication method presented on the Chapter 4, which allows the authentication between them, making an island of devices.

In the context of smart homes, the island of devices can potentiate a business case in which we have an office with several different units in operation. Each of them can have its specific manager/router, but it may be possible to interconnect the different device islands by interconnecting their managers.

There is a manager for each department to authenticate its devices. Different departments can later create a trust alliance between them and can join together to communicate between them safely.

In brief, we can deploy an authentication mechanism for supporting different units inside smart homes, ensuring secure communications end-to-end for all devices authenticated between them.

5.4.3 Summary

Both tests show that it is possible to do a general block and allow only DNS requests trusted by the user.

To improve the implementation and allow users to know the entity belonging to the connection more efficiently, it is sometimes necessary to allow DNS requests that belong to Google but do not have the **.google.com* suffix. For this reason, we implemented an improvement that allows users to understand who a certain domain belongs to (Google, Facebook, or other entities). We add information from ASN that displays much information about IP's but mainly the owner, which users need to understand if it belongs to trusted entities.

5.5 Security Analysis

This section presents an overview of a threat model and some attack scenarios that help to perform a security analysis of our proposed middleware.

Regarding the authentication component, we describe the security analysis and the respective threat model in Section 4.4.

5.5.1 Threat Model

The owner distributes the certificates to trusted devices; therefore, the owner is responsible for that safety component. Assuming that the CA is trusted, all keys signed by the CA will be implicitly trusted. The role of CAs is to ensure that a key belongs to a trusted device after authentication between them. When the IoT device communicates with the router, it knows that it is the real router (as it belongs to the same certification authority) and not an attacker who impersonates it. All communications use HTTPS, and, therefore, all content transmitted between devices is encrypted.

Regarding context-aware access control, and to detect some possible abnormal patterns autonomously, we use the database with a history of usage patterns on each device to reinforce the security parameters by having the user in the loop if there is any unusual behavior. Along with that, if an attacker successfully attacks the system, the user-defined assumptions will take action, and a new HTTPS server that behaves as a router will not be accepted. In brief, a traditional MiTM will not work because the HTTPS certificates will not match. Besides, as already described in Section 5.3.2, the user has to manually intervene when there is a coordination failure in the ambient sensors or based on historical usage patterns.

5.5.2 Attack Scenarios

In this section, we present and evaluate some attack scenarios regarding the system implementation. Section 4.4 already evaluates the tag impersonation and replay attacks regarding the authentication system.

Single Point of Failure

Users can choose the data to be saved on the router or their white-box devices, using a distributed model, rather than stored in a central database or cloud. The advantage of this model, unlike a central database, is that a central database server represents a single point of failure, in which, once compromised, it compromises the security of all users.

We know that we also have a single point of failure on the router, but we have reduced the attack surface because it only compromises one user in an attack. Besides, users can choose not to save data on the router, leaving it only on each of their devices (for a limited time, also defined by the user). With that, we have a distributed model, even inside the user's homes.

5.6 Summary and Future Work

We offer a novel middleware to improve the privacy of users' data on the IoT. The middleware gives users the ability to control the privacy of their data. Users can also store data to be controlled offline and analyze current connections, discarding them according to their preferences, without delay, extensible to network communications.

Unlike previous work, the developed middleware is independent of the device's SDK, as control is placed at the device and router layers, allowing users to control the shared data fully.

We have a real implementation with an RP3 emulating a Google Home with a Google Assistant and sensors attached to it, with the middleware integrated.

As future work, it should be created a real environment with a router with *OpenWRT* running the middleware that we created and with black-box devices together with the RP3s. It is also essential to enhance the current marketplace implementations with aggregated data and anonymization selected by each user, maintaining privacy properties and compliance with TLS.

Chapter 6

The Case for Blockchain in IoT Identity Management

”Thus, the task is not so much to see what no one yet has seen, but to think what nobody yet has thought about that which everybody sees.”

— Arthur Schopenhauer

The IoT has numerous healthcare applications, from remote monitoring to smart sensors and integration of medical devices. The benefits of their adoption can be understood not only in keeping patients safe and healthy, but also in improving care delivery and increasing patient engagement and satisfaction, ensuring that they spend more time interacting with their doctors and responsible.

With the growth of IoT, there is an opportunity to integrate better patient care through home monitoring devices. In addition to being more convenient, home care allows treatment in a family environment, in addition to avoiding the risks of cross-infection that accompany hospital care. Thus, the user would have supervision connected to a central of doctors on duty 24 hours a day, avoiding the aggravation of diseases and reducing the trips to the emergency room. These devices can be integrated with the smart home environment, making it possible to create automatic alerts for falls.

For this care to be carried out safely, the devices need to be authenticated and have a well-defined identity. In these continuous monitoring cases, a lack of proper device identity management can lead an attacker to falsify data, which can lead to patient injury or misdiagnosis.

Current IdM issues that were presented in Section 2.2 can potentially be solved by leveraging blockchain decentralization and resilience [256]. Several works [257–259] claim that the integration between blockchain and IoT should exist to achieve safer and more efficient systems. Blockchain technology can present itself as an alternative to traditional centralized shortcomings by providing a secure solution without a trusted central authority.

This chapter aims to evaluate blockchain's use for IdM in the context of the IoT while focusing on privacy-preserving approaches and their applications to healthcare scenarios. It makes an effort to make a comprehensive description of the most relevant IdM systems focusing on privacy-preserving with or without blockchain and evaluates them against ten selected features grouped into three categories: privacy, availability, and IoT. It is then essential to analyze whether blockchain's use fits all scenarios, according to each feature's importance for different use cases.

In brief, this chapter attempts to explore the following Research Questions:

RQ1: What are the requirements to build a privacy-preserving IdM system for IoT?
RQ2: Can blockchain help us to meet these requirements?
RQ3: What are the impacts of applying blockchain-based IdM systems to healthcare scenarios?

6.1 Related Work

Xiaoyang Zhu and Youakim Badr [139] analyzed traditional IdM systems and investigated new emerging blockchain-based sovereign identity solutions as a way to uncover open challenges in building IdM systems for IoT. The authors argue that privacy remains an open challenge for blockchain because even with the integration of privacy technologies on the public blockchain, these schemes still have an associated complexity. Some examples of these technologies are multiparty computation and zero-knowledge proofs that could bring the selective disclosure and perfect online identity privacy into reality. However, it still needs much effort from academia and industry for their integration with blockchain smart contracts. Another previous work analyzed traditional IdM systems together with privacy-focused IdM systems [128], whereas our approach also focuses on examine whether using blockchain can help satisfying security and privacy, IoT, and usability issues. As such, we analyze blockchain systems and their practical applicability in healthcare scenarios. Healthcare is one of the most dynamically developing and demanding application areas. This application area's exponential growth will affect the medical world, starting from remote health and monitoring services, assisted living, and elderly care to identifying and managing chronic diseases and providing personalized medication [260]. Managing these sensitive and personal data requires preventive measures in data sharing, both privacy and security.

6.2 Building a Privacy-Preserving IdM System

This section will present a comparative table of IdM systems for some features selected according to a selection methodology. We also present a methodology for systems selection.

6.2.1 Methodology for feature selection

This section describes the feature selection criteria that help to evaluate IdM systems.

We chose several features divided among three categories:

- Privacy features: ***Self-Sovereign***, ***Unlinkability***, ***Revocation***, ***Selective Disclosure***, ***Untraceability*** and ***Unforgeability***;
- Availability features: ***Offline***;
- IoT features: ***Device Identity*** and ***Designed for IoT***.

The main feature what we chose to consider to evaluate their suitability towards supporting IoT was ***Device Identity***. Section 2.2 already describes the differences between user and device identity. Device identity will evolve over the next few years with the growth in IoT devices' use. We can see how many people around us carry an activity bracelet, periodically synchronized to a cloud.

Each device in an IoT ecosystem needs a unique device identity as part of an integral part of IoT security. With a unique and strong ***Device Identity***, things can authenticate when they come online and ensure secure and encrypted communication between other devices, services, and users, avoiding possible attacks, such as impersonation attacks. With MiTM, attackers can intercept communications between multiple IoT devices, leading to critical malfunction. For example, a MiTM attack on medical sensors can allow an attacker to change the vital signal's values, leading to misdiagnosis of the patient and wrong treatment.

IoT devices should be able to perform mutual authentication with users or other devices, preferably without the need for central coordination, avoiding SPOF. Devices should perform end-to-end communications in a healthcare context, authenticated with multiple entities, such as other devices or medical assistants, without allowing malicious/unauthorized parties to attack the sensors. With this feature, we will consider that systems that manage a device's identity do not require a user-managed application, allowing devices to communicate independently of the owner.

In addition to the device's identity, it is crucial to understand if the system was ***Designed for IoT***. This concept is paramount as entities must apply technical and organizational measures to ensure correct compliance with IoT requirements throughout the construction process.

Another feature set is focused on privacy-preserving to ensure minimal disclosure of information, inspired by the concept of anonymous credentials [129] (see section 2.2.3).

One of the most critical features for privacy-preserving is ***Unlinkability***, as the adversary should not be able to determine if two blinded credentials belong to the same self-blindable credential [261]. Therefore, this property ensures that different presentations of the same credential cannot be linked [141]. Another important feature inspired by the concept of anonymous credentials is ***Selective Disclosure***. This feature allows users to select only a few attributes necessary to complete authentication in order to protect confidential information from *verifiers* [262]. This method is called *partial identity*. An interesting example is the identification of a person (*prover*) in a movie theater (*verifier*) (see section 2.2.3).

Inspired by the same concept, we selected three more properties: ***Revocation***, ***Untraceability*** and ***Unforgeability***.

Untraceability ensures that a user can display a credential for a *verifier*, without the *verifier* being able to reconstruct the credential back [263].

The concept of anonymous credentials has a solution to protect user's privacy [264]. In case of certificate theft, it is necessary to have ***Revocation*** mechanisms to the suspend or revoke the certificates/identities. Also, users or any *verifiers* may know, within a reasonable time, the revocation state of their credentials [141] to ensure accountability.

The last inspired feature from the concept of anonymous credentials is the ***Unforgeability*** that protects from the forging of certificates [265]. Certificate forging consists of another entity's impersonation; for example, in 2011, an attempted MiTM attack against Google users has been reported. The attacker used a spoofed SSL certificate that impersonated Google's certificate and then received the data between users and Google's encrypted SSL services, such as *Gmail*, *Gdocs*, and others. *DigiNotar* [266, 267] issued more than 500 fraudulent certificates, and 300,000 addresses were compromised. These Google certificates were also issued by this company, which is a legitimate Dutch certification authority that should not issue Google certificates [268]. As a result, a hacker may impersonate someone else to gain access to sensitive information. With the ***Unforgeability*** feature, a malicious user cannot forge certificates.

Inspired by blockchain systems, we also added ***Self-Sovereign*** to the privacy feature set. Individuals no longer depend on a third entity to issue an *identifier* with a self-sovereign identity, as they will create their *identifiers*, maintaining their control and ownership, and the information they wish to share, with whom and under what conditions. The path to a digital identification model without the dependence on an intermediary third party currently translates into the construction of trusted distributed networks, which can validate individuals' information in physical and digital media.

In terms of system availability, one of the most important properties in defining and verifying an identity is the ***Offline***. An example can be unlocking a car with a smartphone where if there is no Internet connection, owners need to take their cars to a location with an Internet connection to download the certificate and establish a connection between the owner's car and phone. Therefore, it is necessary to implement a mechanism to unlock a car even if there is no Internet connection. Researching the concept, we found a paper stating that blockchain can solve this problem [269].

Statistically, we want to compare the ***Commercial/Open Source use*** or number of ***Citations*** of the projects, that also enhance the interest of analysis. ***Commercial/Open Source use*** is based on the number of *GitHub* forks and the number of ***Citations*** is based on Google Scholar (as of 2019-10-16).

6.2.2 Methodology for system selection

This section describes the system selection criteria. First, we need at least two traditional IdM systems to compare with others because it is important to know what kind of resources they can handle, even without focusing on IoT and privacy.

In addition to the traditional IdM systems, we chose two more system sets to compare. The choice of blockchain systems focus on our research questions, and the idea is to analyze whether these systems should be used in IoT towards helping meeting the selected privacy features. For a comparison term, and as there are modern systems that use *Idemix*, it is essential to include the concept of anonymous credentials in the comparison.

The three system groups can be named: Traditional IdM systems, Anonymous credentials-based IdM systems and Blockchain-based IdM systems.

- Traditional IdM systems: ***OpenID*** [270] and ***Shibboleth*** [108, 109];
- Anonymous credentials-based IdM systems: ***SocIoTal Identity Manager*** [128] and ***Anonymous Credential Systems in IoT*** [131];
- Blockchain-based IdM systems: ***UniquID*** [143], ***uPort*** [140], ***Sovrin*** [141] [147], ***Decentralized Privacy-Preserving Healthcare Blockchain for IoT*** [43].

Traditional IdM systems
OpenID is the most widely used [271] and one of the most cited IdM system [270]. ***Shibboleth*** [109] is also one of the most cited among many existing traditional IdM systems. For this reason, we decided to analyze the features of these two systems. Commercial use is not available because the system is closed, and there is no information on how many users or companies are using it.

Anonymous credentials based IdM systems
Due to the features we selected based on the concept of anonymous credentials, that contribute to solving privacy-preserving issues, we discovered two academic research projects (***SocIoTal Identity Manager*** and ***Anonymous Credential Systems in IoT***) using *Idemix* technology and focused on privacy issues and IoT. ***Anonymous Credential Systems in IoT*** has a few citations, but it does include an advantage over ***SocIoTal Identity Manager*** as it is lightweight for IoT, so we think it is essential to include it.

Blockchain-based IdM systems
Finally, we selected four blockchain-based IdM systems. From all the systems analyzed in the section 2.2.4.2, we select those that are more focused on device identity than user identity. *ShoCard*'s

bootstrap uses a trusted identification document (passport or government identification, for example). In these systems, users must manually assign the devices (a user links to a device). Therefore, this system is more focused on user identity than device identity, being farther from an IoT integration than others, so it is not as interesting for our comparison.

UniquID has been selected based on the *Offline* feature and because it is attempting to integrate with IoT technology. We are not only interested in systems designed for IoT, but when we have several blockchain systems to choose from, we preferred those closest to possible integration. ***UniquID*** can solve the *Offline* problem (see section 6.2.1) and focus in IoT, more specifically, IAM of connected things.

uPort and ***Sovrin*** are two interesting systems in the comparison table because both are in the literature about IdM in blockchain. We overview how the current state of integration between IoT and blockchain and its applicability to privacy issues.

Decentralized Privacy-Preserving Healthcare Blockchain for IoT covers three fascinating concepts: healthcare, identity management, and blockchain for IoT. We find it interesting to include in the comparison because, in addition to encompassing the three concepts we are addressing, it also addresses privacy issues and attempts to take a new approach to try to loosen up the blockchain to achieve the desired lightweight for resource-constrained devices.

6.2.3 IdM systems comparison

In this section, we will present a comparison table with the systems we have selected (see section 6.2.2):

1. *OpenID*;
2. *Shibboleth*;
3. *SocIoTal Identity Manager*;
4. *Anonymous Credential Systems in IoT*;
5. *UniquID*;
6. *uPort*;
7. *Sovrin*;
8. *Decentralized Privacy-Preserving Healthcare Blockchain for IoT.*

The first two features of the Table 6.1 show the impact of systems on the community. The first feature is related to commercial usage, while the second is most related to the scientific community. *OpenID* is the most widely used and cited system, according to statistics, but it is also the oldest.

	1	2	3	4	5	6	7	8
Commercial usage (forks)	940 [271]*	N.A.	5[272]	0 [131]	N.A.	56 [273]	184 [274]	N.A.
Citations	596 [270]	225 [109]	11 [128]	N.A.	N.A.	14 [140]	1 [141]	0 [43]
Unlinkability	○	○	●	●	○	○	●	○
Revocation	●	●	●	●	○	●	●	●
Selective Disclosure	○	○	●	●	○	○	●	○
Unforgeability	○	○	●	●	○	○	●	●
Untraceability	●	●	●	●	○	○	●	○
Self-Sovereign	○	○	○	○	○	●	●	●
Offline	○	○	○	○	●	○	○	○
Device Identity	○	○	●	●	●	○	●	○
Designed for IoT	○	○	●	●	●	○	●	●

●Included;
○Not Included.
* The sum of number of forks of different implementations of OpenID. [271]

Table 6.1: IdM systems features comparison

There is no commercial use information in systems 2), 5), and 8) because it is closed source or not yet implemented. All systems have many references and usage, which helped in systems selection because of their relevance. However, systems 4) and 5) do not have much useful information, and citations are scarce. However, the importance of these works is related to their characteristics.

Traditional IdM systems do not cover any of the feature sets.

In terms of IoT features, the authors of *Anonymous Credential Systems in IoT* ensure that the system can satisfy device identity and has the essential features for the integration with IoT. The authors of this paper compare it to *SocIoTal Identity Manager* and create a different lightweight solution because they claim that *SocIoTal Identity Manager* is implemented in Java and has no enhancement to be lightweight. The authors describe that this system is an improvement of *SocIoTal Identity Manager*. Besides, both systems can integrate with IoT.

UniquID and *Sovrin* have device identity, however, we consider that *uPort* does not. The authors state in a *Consensys* [275] presentation that device identity can easily work without a typical username/password combination, but it does not provide a detailed description of how to do it and how advanced it is. *uPort* is not designed for IoT, but the focus of user identity.

In the case of *A Decentralized Privacy-Preserving Healthcare Blockchain for IoT*, we also consider that device identity is not present. Although the system is designed for IoT and is concerned with being lightweight and simplifying multiple protocols to fit few resources, we consider that device identity is not present because the users are responsible for the device management and have healthcare devices attached to them. As far as we know, integrating devices between them requires much effort.

In terms of privacy, anonymous credential-based systems cover much the same and do not cover more blockchain-specific features, such as self-sovereign. *Sovrin* covers almost all the privacy features presented in the table.

6.2.4 Summary of Findings

This section presents a set of features that we need to build a complete privacy-preserving IdM system for IoT (see section 6.2.1), which is the answer to **RQ1**. Then we also present a methodology for selecting systems (see section 6.2.2).

With the results from Table 6.1 (see section 6.2.3), *Sovrin* is the closest production system for IoT with privacy concerns. *Offline* is the only feature *Sovrin* lacks, however this can be adapted because the system is based on blockchain.

With these conclusions, the answer to **RQ2** is yes because *Sovrin* is based on blockchain and address more features than others.

6.3 Discussion on application scenarios

According to the conclusions of the Section 6.2, *Sovrin* is the system that covers the most features presented in the Table 6.1.

There is a research area on IdM for IoT that is not focused on blockchain. Thus, we want to apply an order of importance to the selected features to analyze whether using blockchain is necessary, depending on a specific application case. Therefore, instead of defining whether blockchain can help meet all features, we find it more relevant to define if we should use blockchain in two different IoT scenarios. For this reason, we decided to do a comparative study of two use cases in the healthcare area.

Healthcare is adapting to the IoT paradigm, i.e., a hospital can have IoT sensors and actuators spread across some rooms and patients. Several solutions give patients the ability to have more control over their health, educating them about the importance of maintaining a healthy diet and frequent exercise. An example of a wearable device is the *Fitbit*, which monitors signals, such as heart rate and sleep quality. Also, there are simple health apps on smartphones that measure the number of steps, for example. By continually receiving medical data, patients become more aware of their physical condition and better prepared to take care of themselves.

It is not the first time we have seen the news from activity trackers that have made a difference in health and disease detection, such as Apple Watch, which warned an 18-year-old girl that her resting heart rate was 190 beats per minute and that she must seek immediate help [276].

From pacemakers to blood pressure cuffs, IoT health services can also help doctors better manage disease, monitor patients, and improve treatment outcomes – but the security of health data is a substantial risk.

Identity theft can be one of IoT's most significant issues, as one device can impersonate another to receive privileged information or bypass access controls. Addressing this situation is critical as it endangers the health and privacy of patients.

We choose different healthcare scenarios to define different identity and security needs. We will define a scenario that fits intensive care (where treatment should be urgent and instantaneous) and continuous care (treatment is continuous and not instantaneous. Patients' health data is shared with doctors to let them know their health overtime for additional assertive treatment based on more data).

6.3.1 Intensive Care Medicine

Generally, patients in an Intensive Care Unit (ICU) need careful observation and need intense and constant concern. As patients in this situation have suffered severe injuries or have recently undergone surgery, it is critical to monitor patients with sensors attached to their bodies to detect their vital signs. If there is a problem, alarms should be issued to inform a doctor that a patient needs treatment [277].

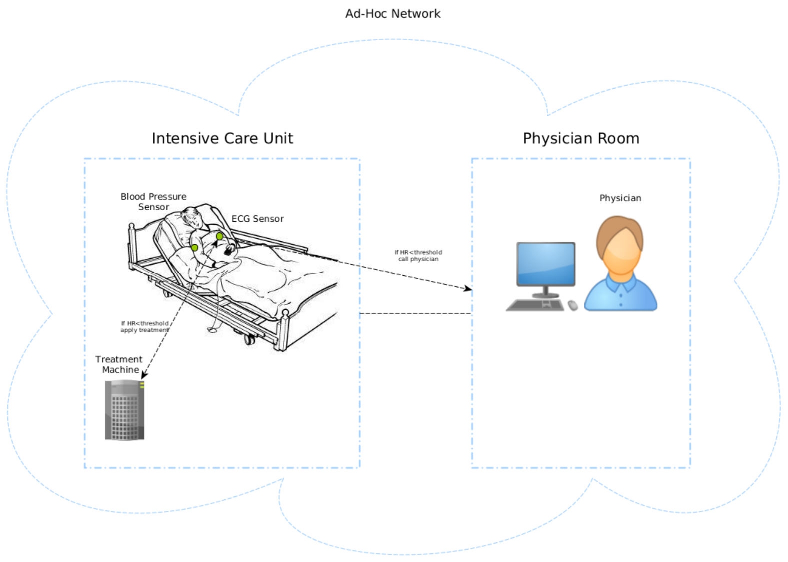

Figure 6.1: Intensive Care Healthcare
[278]

We describe an identical scenario in which a patient is being treated only with a machine's aid based on IoT technologies [279]. Thus, there is no human presence to assist patients and machines. In this scenario, a control room should control a set of roles and trust embodiment, and surgery data is maintained and analyzed based on patient recovery, for example, in a recovery room (Figure 6.1 represents the scenario). For example, if a patient has a heart rate below a threshold, the room detects this autonomously and alerts the attending or nearest medical doctor. However, at the same time, room actuators may undergo some patient treatment, such as Basic Life Support (BLS).

These systems are critical because, based on surgery and the postoperative methodology, the systems may be different even for surgeries with similar people (age, gender, size) and may change according to similar facts seen in the surgery room. These tasks should be learned from previous data mining rules and fed with the most up-to-date information collected from the patient.

This scenario may suffer impersonation issues because an attacker could falsify patient data and the room's heart rate sensors, causing machines connected to the sensors (actuator) to provide unnecessary treatments. For example, a BLS can cause injury to patients.

We list the features in order of importance for this specific application scenario:

1. **Offline** - Offline is the most important feature in this scenario because if there is a failure on the Internet, it makes no sense to interfere with the warning of abnormal heart rate values, which can cause serious harm to the patient;

2. **Device Identity** - With a device identity, things can be authenticated with each other or

with humans. This feature is important primarily because of device lifecycle management, authentication, and access control. For example, if there is no authentication, an attacker's heart rate sensor can impersonate a real sensor and produce erroneous values, sending dispatch alarms to doctors and inferring incorrect remedies for the patient;

3. **Unforgeability** - This feature is necessary because if a device with a sensor is susceptible to a Sybil attack, it may mislead the treatment;

4. **Self-Sovereign** - Along with the device identity feature, decentralization is important to prevent the attacker from exploiting a central point of failure.

5. **Revocation** - IoT devices may have a long lifespan, but some are disposable sensors. Therefore, it is necessary that at any time during the period of use, the system may define a sensor as revoked;

6. **Selective Disclosure** - IoT devices should only disclose relevant information about owners for communication with other parties. However, it may be essential to access other details about the patient. For this reason, and if necessary, by the doctor, it may be possible to break security and access data;

7. **Unlinkability/Untraceability** - These are the least important features for this healthcare scenario. It may be essential to know the owner of a sensor, or different sensors, for an urgent treatment situation. Systems should integrate these features by default, but it must be possible to break the security (during an urgency) and access data.

6.3.2 Continuous Healthcare

Continuous healthcare is a scenario that can prevent some fatal cases due to early detection and rapid response [280]. Health devices, such as ECG monitors, glucose monitors, pulse oximeters, and blood pressure monitors, exist and will soon integrate with micro and nano-chemical sensors, which will provide ongoing medical diagnostics [281] (Figure 6.2 represents this scenario). This type of healthcare can help detect some illnesses through medical monitoring (e.g., diabetic patients, other metabolic diseases, skin diseases, and drug pharmacokinetics).

In this kind of scenario, the treatment is continuous. Medical doctors receive patients' health data for continuous surveillance and more assertive treatment based on more data. Generally, patients requiring this type of treatment are chronic or healthy patients undergoing continuous treatment or are continuously monitored, just in case. That way, if patients have a 24-hour tracking service every day, it will help doctors adjust treatment according to their health condition.

As already described, some wearable devices, such as *Fitbit* or *Apple Watch*, help in continuous monitoring of a patient, even being healthy. More recently, *Fitbit* and *Google* have invested in healthcare, and the two have teamed up to improve the way fitness trackers handle their users' health

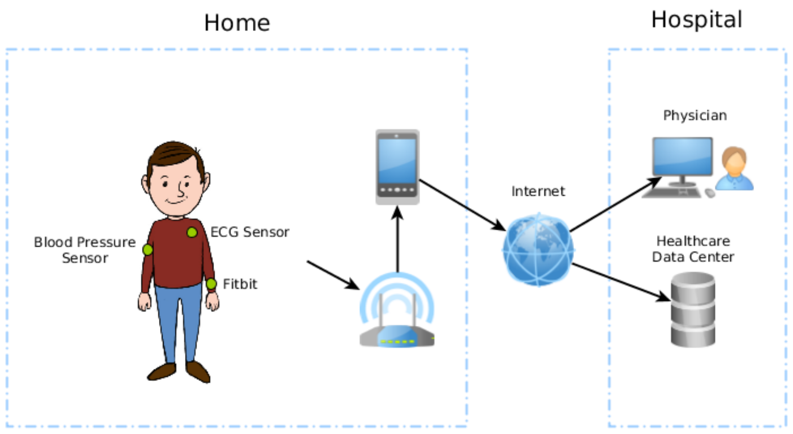

Figure 6.2: Continuous Healthcare
[282]

data. According to some news [283], a fitness manufacturer announced that it would implement the *Google Healthcare API* to provide health data directly to the electronic medical record systems used by doctors and hospitals. The union aims to provide healthcare professionals with more detailed patient information to receive personalized, efficient, and tailored care. Thus, companies expect that this technology will help doctors diagnose a disease and respond effectively with this constant stream of user-health data, which is an evolution of this technology, as in recent years it has been used only to track user well-being and activity or to be used by personal trainers to view activity and track athlete's data.

To classify features in this application scenario, we use a different methodology because we think all privacy features have the same importance, and we also need to have the device identity. The only feature we consider less critical would be offline, which belongs to the availability feature set. Unlike the previous scenario, offline is not a high priority property, and continuous care does not require offline communication in case of Internet failure, as it is not critical to act immediately. In this scenario, alerts and data taken from the patient's health are stored and sent to the medical doctor later.

6.3.3 Conclusions about the use of blockchain in healthcare

According to these scenarios, this section analyzed different healthcare scenarios and prioritized features to answer to **RQ3**. However, according to the results, the answer is: it depends on the application scenario.

In IoT healthcare, in general, it may be beneficial to use blockchain in IoT scenarios, as it can also provide more reliable and secure strategies for an IoT medical device to promote the Identity of Medical Things applications and smart contracts to automate device life cycle management [42].

As healthcare works with a lot of sensitive user information, blockchain can help with that. It is also essential to use blockchain to eliminate the SPOF problems and allow users to control all their information and transactions. It provides secure access, scalability, and data privacy, allowing a greater agility in transaction records, traceability of medical supplies (such as medicines, orthotics, and prostheses), or sharing data from patients to reduce repeated examinations.

Intensive Care Medicine

More specifically, in intensive care, we can say that it is unnecessary to use *Sovrin*, for example, because *UniquID* has the most appropriate features to prioritize and is also the only one that has the offline feature. However, this system neglects some privacy issues, which also concern user data. While privacy features are minor in this application case, maintaining them is still vital.

UniquID system may integrate a system based on *Idemix*, such as *Sovrin* does, and may have the properties of selective disclosure, unlinkability, and untraceability. However, the medical doctor must break this security for a larger purpose in emergency scenarios. This mechanism is called *break the glass*, which means that users can gain access to some information in an emergency even if they are not allowed to access it. This option should be given to the medical doctor to access patient data in an emergency (assuming the risk of continuing and recording such action). The inclusion of this option is an exciting future research direction to apply to these systems or equivalent.

Besides, traceability is essential to detect machine failures that can cause patient injury. We need to ensure that there is access to the device's data, such as vendor name, model and serial number, version, physical location, support contacts, or any other relevant data points that can help specialists to quickly identify the device (device identity) and minimize the damage that can occur in the worst-case scenario. We do not want a malfunctioning machine to be in a future surgery room because we completely anonymized the information and can no longer track the device.

Continuous Healthcare

In a continuous care healthcare setting, systems must archive data so that users can consult later. It is unnecessary to have all the urgency-related features, such as the offline property that guarantees access to data even without an Internet connection. Therefore, this is not a priority.

Subsequently, we rank the order of priority according to the purpose of privacy. In this case of application, it is more important to ensure personal data privacy and anonymity, as it is often an ongoing treatment where critical health data is not needed (or what is needed is already being shared). In the case of a personal trainer, it is unnecessary to access data, such as the patient's blood type.

Therefore, we assume that, as far as we know, ***Sovrin*** is the most completed system. However, it would be unfair if we did not say that we could also choose system 4, as only the feature of self-sovereignty is lacking. In this case, the feature is essential, but once we have features, such as unlinkability and selective disclosure, which ensure data privacy and anonymity, we can loosen up the model.

Final decision
Regarding the question **RQ3**, the section 6.3.3 begins with the answer: it depends on the application scenario, and the final decision follows the same parameters even after evaluating each use case.

Sovrin and ***UniquID*** were the main systems chosen because of their characteristics. However, if some developers do not want to use blockchain to build a privacy-preserving IdM implementation for healthcare devices, because it is expensive in terms of computing and often involves high costs and width delays, which are not directly suited for most IoT devices, developers may adopt different solutions [42].

From these analyses, we conclude that the most important in these systems are the privacy features. The systems ”SocIoTal Identity Manager” and ”Anonymous Credential Systems in IoT” have solutions based on the concept of anonymous credentials that include these privacy features.

6.4 Limitations and Future Research Directions

After analyzing and answering the research questions presented in this chapter, more questions have presented themselves to improve this research area.

1) Implementing break the glass mechanisms or policies to gain access to required user data: It is necessary to implement *break the glass* mechanisms to enhance an IdM system, complementing the features we have chosen. Privacy features must always be present and implemented, regardless of the use case. The most common use of a *break the glass* mechanism is related to fire alarms, where a user must break a thin glass layer before doing anything, discouraging accidental activation. This concept already exists in healthcare and is associated with opening confidential records and being recorded on an accounting disclosure form. In this paper, we provide an example of using *break the glass* in an intensive care use case (see section 6.3.3), but may apply to other use cases. As far as we know, there is not yet an IdM system with privacy features that do this, and we believe it would be an improvement and a significant innovation to the proposed requirements for building a complete IdM system for IoT in order to guarantee privacy preservation.

2) Context-based Device Identity: It is of utmost importance to consider managing IoT identity and authentication, as multiple entities need to authenticate with each other to create and provide trustable services. As billions of things will be interconnected, identity management needs a highly scalable way. The identity of things should be based not only on their attributes but also on their **context**. We have identified some systems with context-based identity; however, many of them have context-based only on attributes, policies, or rules. Context can define different identities for a device and access control policies. Context attributes include the environment in which the device is, for example, its location. Users who have access to wearable technology (such as fitness trackers) may choose to use smart lock technology to unlock their smartphones through a set of trusted devices when connected via Bluetooth. However, users may want to enable this technology only at home as it is a controlled environment, and only the family can mess with their smartphone and turn it off at work. Users would have different access control for each user, depending on the users' location.

3) Device Identity without depending on an owner: Given the context's introduction, it is also necessary to implement proprietary device identities without relying on that device's owner. It would be interesting to manage a device identity to have multiple identities, depending on the context, with different and multiple *personas*. Currently, IdM systems have some of these properties, but it always depends on the owner of a sensor, and it would be interesting to eliminate this dependency.

4) Anonymous Credential concept on IoT devices: After this definition of identity, it would be exciting to integrate the properties of the concept of Anonymous Credential into the device. These properties apply to the device owner's data, but it is crucial to do this privacy control of device properties and data.

5) Standard Blockchain-based solutions for IoT: Blockchain presents many promising opportunities for the future of IoT. The challenges, however, remain in the form of consensus models and computational transaction verification costs. Using blockchain would be possible in all scenarios. However, to make the systems usable and less computationally intensive, as suggested by [111] in IoT, systems should be as simple and as small as possible, but without decreasing security levels/properties. Although there are already some approaches to integrating blockchain with IoT, a default definition is required.

6.5 Summary

IoT has become a common household name with its continuous and ever-growing adoption in our daily lives, fueled mainly by the convenience and efficiency it brings. However, the vast amounts of data it generates, including sensitive information, require more fail-safes than the currently employed, e.g., proper identity management is required to prevent impersonation attacks.

There are several IdM approaches to IoT, with and without blockchain. We have seen that blockchain can help resolve centralizing and registering legitimate nodes by allowing devices to identify and authenticate themselves.

Given the postulated research questions, we have determined a set of features needed to define an IoT-focused privacy-preserving IdM system. Although blockchain has many benefits, we conclude it should not be used as a panacea for all scenarios.

As a culmination of this work, different future research questions were brought related to creating a device IdM system, drive by the need for a device having to have an owner and be unable to have an identity by itself. Besides, context-based identity can enhance better management of authentication and access controls.

To improve IdM systems, we propose implementing privacy features by default, including the properties of the concept of anonymous credentials. Also, the integration of break the glass mechanisms that can break associated security mechanisms for a greater good, even in sensitive healthcare settings where medical doctors may need to access some encrypted confidential patient information, is essential.

Regarding blockchain's usage, there is a substantial amount of ongoing research with different solutions for integrating this technology with IoT. However, a standard solution that deals with the complexity and adaptability of resource-limited devices is needed.

Chapter 7

Conclusions

”In a world where everyone is overexposed, the coolest thing you can do is maintain your mystery.”

— Unknown

This thesis designs and implements a privacy-preserving middleware focused on keeping users’ data protected and controlled. This middleware focuses on integrating several components, namely end-to-end authentication for devices (possible to scale), the network traffic security solution to control device communications with the Internet, and configurable privacy policies for users.

Our approach was motivated by the lack of support for users to control device data sharing and internal privacy policies. We also want to have context-based privacy preferences to target different ”personas” to a device, behaving differently (with different permissions) according to the environment. There are no middleware solutions that integrate all the components presented in this thesis, as far as we know.

In this thesis, we started the middleware project and the integration of the components. This entire system was evaluated, not only with the Raspberry PI, but also with devices dependent on the manufacturer, in this case, Google Home. Performance tests show that middleware is efficient for routers and white box devices like the Raspberry PI.

The authentication component also shows positive results, both concerning the latency times and the battery.

We also run security tests and a threat model to prevent and assess all components’ resistance to multiple attackers.

The results of the assessment support the idea that the approach taken is valid.

7.1 Contributions

This thesis has produced the following contributions:

Literature Review: Chapter 2 contains a current and past status on MPC, and its combination with Blockchain as a way to report on one of the most well-known privacy-preserving techniques in fog computation. Then, we provide a complete review of the concepts of identity management, secure point-to-point communications, device provisioning, network traffic solutions, and middleware solutions for IoT.

The Case for Blockchain in IoT Identity Management: We present an evaluation of the use of Blockchain for IdM in the context of the IoT in Chapter 3. It focuses on privacy-preserving approaches and their applications to healthcare scenarios. Chapter 3 contributes with new research lines for IdM systems in IoT, including challenges related to device identity definition, privacy-preserving, and new security mechanisms.

End-to-End Secure Communications
Chapter 4 evaluates one of the privacy-preserving techniques addressed in Chapter 2 and combines two existing concepts for end-to-end communications (namely, ZRTP and J-PAKE). We realized that MPC does not guarantee authentication without an authenticated channel during development. For this reason, we adapted the solution by combining the concepts of ZRTP and J-PAKE with a channel generated by the LLC concept with infrared. Although it is not the focus of this thesis, this chapter also contributes to the usability of ZRTP because, with MPC, there is no longer the need to manually validate the keys of each smartphone (in this case, the SIP channel authenticates the MPC communications). Regarding IoT, this chapter opens research lines on the scalability of the solution's solution and identity management with LLC (it is a solution that works but is very limited, only for 1 to 1 authentication in controlled houses).

Secure Provisioning for Achieving End-to-End Secure Communications
We address device authentication/provisioning, especially regarding some research limitations, namely, device identity, scalability, and resistance to MiTM. Chapter 5 presents a solution that relies on the combination of a secure token (capable of generating OTP and storage of a PKI) with cryptographic algorithms on a hybrid authentication system. After device provisioning, the system can provide a decentralized architecture where the trusted devices can communicate peer-to-peer. This system's main advantage is the use of ephemeral keys for clients and the manager's offline cryptographic assets, including storing the private key isolated from network access and kept in a powered-down state. Here, the usage of a secure hardware token is the best solution for that.

Empowering Users Through a Privacy Middleware Watchdog
Finally, Chapter 6 contributes to the design and construction of the privacy middleware structure, integrating device identity and secure communication between the devices, which was the initial goal of this thesis. This chapter presents the first component that effectively controls the network data that the devices generate. Along with the network control, techniques for users' control of their data locally and offline are also addressed while preserving their privacy. This chapter presents the overall middleware implementation and integration into two use cases, including real IoT devices, such as Google Home and Phillips Hue, to evaluate its effectiveness and feasibility. This chapter opens new investigation lines, especially concerning access control (neighbors or visitors) and the improvement of context-based identity.

7.2 Limitations and Future Research Challenges

Considering the security analysis and results provided in this thesis, it is possible to conclude that the task of provisioning/authentication of devices is not trivial. The integration was done in middleware to protect users' privacy, but there are still many things to improve.

There are many research challenges highlighted in the Section 6.4 regarding the Identity Management in IoT.

This section focus on the limitations and future research directions of our proposed solutions. Nevertheless, as expected, the work presented in this thesis also provides some limitations, and therefore, suggestions for future work.

Semi-automatic Privacy Policies
Traditional security mechanisms use an unconscious approach, using static parameters to provide individual decisions. However, IoT is a dynamic environment. Real-world applications combine computing, communication, detection, and, in some cases, performance to monitor and control the environment remotely. We must take advantage of the resources and capacity that the IoT has and incorporate context-sensitive resources at the lowest level (hardware) and in the software layers. Context-aware computing played an essential role in addressing this challenge in previous paradigms, such as generalized computing, which leads us to believe that it would continue to be successful in the IoT paradigm. In an environment like the Internet of Things, where billions of objects communicate, a significant amount of energy can be saved, following straightforward optimization techniques, for example.

Recognizing relevant contextual attributes and using them to automatically turn off some voice permissions is quite interesting as it improves the usability and privacy of all users in multiple contexts. It is challenging to develop privacy models that capture all the possible scenarios, but we can adjust the privacy settings to be turned off by default, such as in the previous sections, allowing only explicit selected permissions.

The contextual attributes can detect different environments where the user is. If the middleware detects a mobile device inside the same room as the IoT device recording voice, and if this user does not have to accept the microphone's permission, it cannot record any voice since this time.

It is challenging to have context for all situations (for example, the presence of a child) without violating even worse privacy rights, but since it blocks permissions by default, privacy is more adapted to the situation of the presence of strangers.

Information Dissemination

Chapter 5 introduces the certification management and information dissemination, but creating distributed system algorithms is not in this thesis's scope. This thesis assumes a mobile device that disseminates information or fixed points to disseminate the information at the end of the day (or at fixed times of the day). However, it is of utmost importance to create algorithms capable of doing that efficiently.

Internal Malicious Nodes

In Chapter 5, after a device joins a pool of trusted devices, there is no further surveillance to see if that node is reliable or not. A reputation system on the network could help solve this problem, giving a ranking according to the device's behavior.

Information Flow Control

IoT has numerous applications today, and its use is increasing everywhere. Generating, processing, analyzing, sharing, and storing large amounts of actionable data also raises several concerns and challenges. Privacy and data protection issues also arise with creating smart cities. Many smart cities technologies capture Personally Identifiable Information (PII), and family-level data about citizens - their characteristics, location and movements, and their activities - link this data to produce new derived data and create profiles of people and places to make decisions about them.

It is essential to ensure mechanisms that allow controlling the dissemination of this data.

Data privacy protection is a topic of intense research due to the increasing amount of data transmitted over the network and manipulated by untrusted entities. Although security methods such as firewalls and encryption prevent data from being accessed in an unauthorized manner, they do not guarantee that this data will not be improperly propagated after access. For example, encryption allows data to be exchanged over an insecure channel, but it does not guarantee that data confidentiality after being decrypted on the receiving side. Information Flow Control (IFC) promises to tackle this problem by analyzing the flow of information within the system, assigning security levels to data and entities that manipulate it.

Having these components completed, we need to tackle the information flow between devices because the dissemination needs to be private. It is essential to control the life cycle of the information flow.

Private keys stored on the device

The secure provisioning approach presented in this thesis has a significant challenge. Devices become an active part of security by being responsible for keeping their private keys safe. One possible solution is to use the SGX technology for key protection [284].

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of things (iot): A vision, architectural elements, and future directions,” *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013. [Cited on page 7.]

[2] N. P. R. . E. Research, “The smart audio report,” *available at: https://n.pr/2zEk4UE (Accessed 11 December 2020)*, 2017. [Cited on page 8.]

[3] J. Ren, D. J. Dubois, D. Choffnes, A. M. Mandalari, R. Kolcun, and H. Haddadi, “Information exposure from consumer iot devices: A multidimensional, network-informed measurement approach,” in *Proceedings of the Internet Measurement Conference*, 2019, pp. 267–279. [Cited on page 8.]

[4] M. Goddard, “The eu general data protection regulation (gdpr): European regulation that has a global impact,” *International Journal of Market Research*, vol. 59, no. 6, pp. 703–705, 2017. [Cited on page 8.]

[5] J. Lopez, R. Rios, F. Bao, and G. Wang, “Evolving privacy: From sensors to the internet of things,” *Future Generation Computer Systems*, vol. 75, pp. 46–57, 2017. [Cited on page 8.]

[6] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, “Evaluating critical security issues of the iot world: Present and future challenges,” *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2483–2495, 2017. [Cited on page 9.]

[7] M. A. Khan and K. Salah, “Iot security: Review, blockchain solutions, and open challenges,” *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018. [Cited on page 9.]

[8] H. Aksu, L. Babun, M. Conti, G. Tolomei, and A. S. Uluagac, “Advertising in the iot era: Vision and challenges,” *IEEE Communications Magazine*, vol. 56, no. 11, pp. 138–144, 2018. [Cited on page 9.]

[9] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, “A survey on security and privacy issues in internet-of-things,” *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1250–1258, 2017. [Cited on page 9.]

[10] M. Conti, A. Dehghantanha, K. Franke, and S. Watson, "Internet of things security and forensics: Challenges and opportunities," 2018. [Cited on page 9.]

[11] H. Wang, Z. Zhang, and T. Taleb, "Special issue on security and privacy of iot," *World Wide Web*, vol. 21, no. 1, pp. 1–6, 2018.

[12] C. Yang, Q. Huang, Z. Li, K. Liu, and F. Hu, "Big data and cloud computing: innovation opportunities and challenges," *International Journal of Digital Earth*, vol. 10, no. 1, pp. 13–53, 2017. [Cited on page 9.]

[13] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010. [Cited on page 9.]

[14] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad hoc networks*, vol. 10, no. 7, pp. 1497–1516, 2012. [Cited on page 9.]

[15] P. A. Laplante and N. Laplante, "The internet of things in healthcare: Potential applications and challenges," *It Professional*, vol. 18, no. 3, pp. 2–4, 2016. [Cited on page 9.]

[16] M. Zhang, "If your iphone has a green dot in ios 14, your camera may be spying on you." available at: https://bit.ly/3bnNtkO (Accessed 20 February 2021), 2020. [Cited on page 9.]

[17] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13–16. [Cited on page 15.]

[18] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game, or a completeness theorem for protocols with honest majority," in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, 2019, pp. 307–328. [Cited on page 15.]

[19] A. C. Yao, "Protocols for secure computations," in *23rd annual symposium on foundations of computer science (sfcs 1982)*. IEEE, 1982, pp. 160–164.

[20] A. C.-C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE, 1986, pp. 162–167. [Cited on page 15.]

[21] G. Zyskind, O. Nathan, and A. Pentland, "Enigma: Decentralized computation platform with guaranteed privacy," *arXiv preprint arXiv:1506.03471*, 2015. [Cited on pages 16 and 29.]

[22] J. Yli-Huumo, D. Ko, S. Choi, S. Park, and K. Smolander, "Where is current research on blockchain technology?—a systematic review," *PloS one*, vol. 11, no. 10, p. e0163477, 2016. [Cited on page 16.]

[23] K. Wüst and A. Gervais, "Do you need a blockchain?" in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT).* IEEE, 2018, pp. 45–54. [Cited on page 16.]

[24] M. Mainelli and M. Smith, "Sharing ledgers for sharing economies: an exploration of mutual distributed ledgers (aka blockchain technology)," *Journal of Financial Perspectives*, vol. 3, no. 3, 2015. [Cited on page 16.]

[25] M. Nuss, A. Puchta, and M. Kunz, "Towards blockchain-based identity and access management for internet of things in enterprises," in *International Conference on Trust and Privacy in Digital Business*. Springer, 2018, pp. 167–181. [Cited on page 16.]

[26] N. Garg, *Apache Kafka*. Packt Publishing Ltd, 2013. [Cited on page 16.]

[27] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, 1982. [Cited on page 17.]

[28] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE International Congress on Big Data (BigData Congress).* IEEE, 2017, pp. 557–564. [Cited on page 17.]

[29] D. Ron and A. Shamir, "Quantitative analysis of the full bitcoin transaction graph," in *International Conference on Financial Cryptography and Data Security*. Springer, 2013, pp. 6–24. [Cited on page 17.]

[30] M. Swan, *Blockchain: Blueprint for a new economy*. O'Reilly Media, Inc., 2015.

[31] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019. [Cited on page 17.]

[32] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains," in *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, 2016, pp. 181–194. [Cited on page 17.]

[33] P. Swathi, C. Modi, and D. Patel, "Preventing sybil attack in blockchain using distributed behavior monitoring of miners," in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT).* IEEE, 2019, pp. 1–6. [Cited on page 17.]

[34] H. F. Atlam, A. Alenezi, M. O. Alassafi, and G. Wills, "Blockchain with internet of things: Benefits, challenges, and future directions," *International Journal of Intelligent Systems and Applications*, vol. 10, no. 6, pp. 40–48, 2018. [Cited on page 17.]

[35] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, "On blockchain and its integration with iot. challenges and opportunities," *Future generation computer systems*, vol. 88, pp. 173–190, 2018. [Cited on page 18.]

[36] N. Kshetri, “Can blockchain strengthen the internet of things?” *IT professional*, vol. 19, no. 4, pp. 68–72, 2017. [Cited on page 18.]

[37] A. Dorri, S. S. Kanhere, and R. Jurdak, “Blockchain in internet of things: challenges and solutions,” *arXiv preprint arXiv:1608.05187*, 2016. [Cited on pages 18 and 19.]

[38] T. M. Fernández-Caramés, O. Blanco-Novoa, I. Froiz-Míguez, and P. Fraga-Lamas, “Towards an autonomous industry 4.0 warehouse: A uav and blockchain-based system for inventory and traceability applications in big data-driven supply chain management,” *Sensors*, vol. 19, no. 10, p. 2394, 2019. [Cited on page 18.]

[39] S. Dawaliby, A. Aberkane, and A. Bradai, “Blockchain-based iot platform for autonomous drone operations management,” in *Proceedings of the 2nd ACM MobiCom Workshop on Drone Assisted Wireless Communications for 5G and Beyond*, 2020, pp. 31–36. [Cited on page 18.]

[40] K. J. O’Dwyer and D. Malone, “Bitcoin mining and its energy footprint,” *25th IET Irish Signals & Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014)*, 2014. [Cited on page 18.]

[41] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, “Lsb: A lightweight scalable blockchain for iot security and privacy,” *arXiv preprint arXiv:1712.02969*, 2017. [Cited on page 18.]

[42] A. Dorri, S. S. Kanhere, and R. Jurdak, “Towards an optimized blockchain for iot,” in *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*. ACM, 2017, pp. 173–178. [Cited on pages 18, 108, and 110.]

[43] A. D. Dwivedi, G. Srivastava, S. Dhar, and R. Singh, “A decentralized privacy-preserving healthcare blockchain for iot,” *Sensors*, vol. 19, no. 2, p. 326, 2019. [Cited on pages 18, 38, 40, 101, and 103.]

[44] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, “When mobile blockchain meets edge computing,” *IEEE Communications Magazine*, vol. 56, no. 8, pp. 33–39, 2018. [Cited on page 19.]

[45] K. Christidis and M. Devetsikiotis, “Blockchains and smart contracts for the internet of things,” *Ieee Access*, vol. 4, pp. 2292–2303, 2016. [Cited on page 19.]

[46] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, “Bitcoin-ng: A scalable blockchain protocol,” in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, 2016, pp. 45–59. [Cited on page 19.]

[47] E. A. Brewer, “Towards robust distributed systems,” in *PODC*, vol. 7, 2000. [Cited on page 19.]

[48] K. Yeow, A. Gani, R. W. Ahmad, J. J. Rodrigues, and K. Ko, “Decentralized consensus for edge-centric internet of things: A review, taxonomy, and research issues,” *IEEE Access*, vol. 6, pp. 1513–1524, 2017. [Cited on page 19.]

[49] S. H. Hashemi, F. Faghri, P. Rausch, and R. H. Campbell, “World of empowered iot users,” in *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2016, pp. 13–24. [Cited on page 19.]

[50] H. Sukhwani, J. M. Martínez, X. Chang, K. S. Trivedi, and A. Rindos, “Performance modeling of pbft consensus process for permissioned blockchain network (hyperledger fabric),” in *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2017, pp. 253–255. [Cited on page 19.]

[51] M. Milutinovic, W. He, H. Wu, and M. Kanwal, “Proof of luck: An efficient blockchain consensus protocol,” in *proceedings of the 1st Workshop on System Software for Trusted Execution*. ACM, 2016, p. 2. [Cited on page 19.]

[52] A. Panarello, N. Tapas, G. Merlino, F. Longo, and A. Puliafito, “Blockchain and iot integration: A systematic survey,” *Sensors*, vol. 18, no. 8, p. 2575, 2018. [Cited on page 19.]

[53] J. Zou, B. Ye, L. Qu, Y. Wang, M. A. Orgun, and L. Li, “A proof-of-trust consensus protocol for enhancing accountability in crowdsourcing services,” *IEEE Transactions on Services Computing*, 2018. [Cited on page 19.]

[54] S. Roy, M. Ashaduzzaman, M. Hassan, and A. R. Chowdhury, “Blockchain for iot security and management: Current prospects, challenges and future directions,” in *2018 5th International Conference on Networking, Systems and Security (NSysS)*. IEEE, 2018, pp. 1–9. [Cited on page 19.]

[55] M. Divya and N. B. Biradar, “Iota-next generation block chain,” *International journal of engineering and computer science*, vol. 7, no. 04, pp. 23 823–23 826, 2018. [Cited on pages 20 and 21.]

[56] J. S. Preden, K. Tammemäe, A. Jantsch, M. Leier, A. Riid, and E. Calis, “The benefits of self-awareness and attention in fog and mist computing,” *Computer*, vol. 48, no. 7, pp. 37–45, 2015. [Cited on page 20.]

[57] G. Bohl and J. F. Dickson, “Private blockchains in automotive safety,” 2017. [Cited on page 20.]

[58] W. F. Silvano and R. Marcelino, “Iota tangle: A cryptocurrency to communicate internet of things data,” *Future Generation Computer Systems*, 2020. [Cited on page 21.]

[59] K. Dasgupta and M. R. Babu, “A review on crypto-currency transactions using iota (technology),” in *Social Network Forensics, Cyber Security, and Machine Learning*. Springer, 2019, pp. 67–81. [Cited on page 21.]

[60] P. Veena, S. Panikkar, S. Nair, and P. Brody, "Empowering the edge-practical insights on a decentralized internet of things," *IBM Institute for Business Value*, vol. 17, 2015. [Cited on page 21.]

[61] J. M. Cohn, P. G. Finn, S. P. Nair, S. B. Panikkar, and V. S. Pureswaran, "Autonomous decentralized peer-to-peer telemetry," Apr. 9 2019, uS Patent 10,257,270. [Cited on page 21.]

[62] M. Signorini *et al.*, "Towards an internet of trust: issues and solutions for identification and authentication in the internet of things," Ph.D. dissertation, Universitat Pompeu Fabra, 2016. [Cited on page 21.]

[63] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019. [Cited on page 21.]

[64] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *self-published paper, August*, vol. 19, p. 1, 2012. [Cited on page 21.]

[65] J. Miller, "Telehash: Encrypted mesh protocol." available at: https://github.com/telehash/telehash.github.io (Accessed 11 December 2020), 2013. [Cited on page 21.]

[66] J. J. Karst and G. Brodar, "Connecting multiple devices with blockchain in the internet of things," in *Res. Paper Seminar Blockchain Technol.(MTAT. 03.323), Tech. Rep*, 2017. [Cited on page 21.]

[67] A. C. Yao, "Protocols for secure computations," in *23rd annual symposium on foundations of computer science (sfcs 1982)*. IEEE, 1982, pp. 160–164. [Cited on pages 22, 23, and 24.]

[68] M. Hirt, U. Maurer, and B. Przydatek, "Efficient secure multi-party computation," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2000, pp. 143–161. [Cited on page 22.]

[69] A. C. Yao, "Theory and application of trapdoor functions," in *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*. IEEE, 1982, pp. 80–91. [Cited on page 22.]

[70] Y. Lindell, "Secure multiparty computation for privacy preserving data mining," in *Encyclopedia of Data Warehousing and Mining*. IGI Global, 2005, pp. 1005–1009. [Cited on page 22.]

[71] P. Pullonen and S. Siim, "Combining secret sharing and garbled circuits for efficient private ieee 754 floating-point computations," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 172–183. [Cited on page 22.]

[72] S. Havron, "Poster: Secure multi-party computation as a tool for privacy-preserving data analysis," *Semantic Scholar*, 2016. [Cited on page 25.]

[73] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," in *European Symposium on Research in Computer Security*. Springer, 2008, pp. 192–206. [Cited on pages 25 and 27.]

[74] Cybernetica, “Sharemind sdk,” available at: https://sharemind-sdk.github.io/ (Accessed 11 December 2020), 2015. [Cited on pages 25 and 27.]

[75] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, “Multiparty computation from somewhat homomorphic encryption,” in *Annual Cryptology Conference*. Springer, 2012, pp. 643–662. [Cited on page 26.]

[76] M. K. et al., “Multiparty computation with spdz, mascot, and overdrive offline phases,” available at: https://github.com/bristolcrypto/SPDZ-2 (Accessed 11 December 2020), 2016. [Cited on page 26.]

[77] M. Keller, “Mp-spdz: A versatile framework for multi-party computation,” available at: https://github.com/data61/MP-SPDZ (Accessed 11 December 2020), 2020. [Cited on pages 26 and 27.]

[78] ——, “Mp-spdz: A versatile framework for multi-party computation.” *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 521, 2020. [Cited on pages 26 and 27.]

[79] D. Malkhi, N. Nisan, B. Pinkas, Y. Sella *et al.*, “Fairplay-secure two-party computation system.” in *USENIX Security Symposium*, vol. 4. San Diego, CA, USA, 2004, p. 9. [Cited on page 26.]

[80] A. Ben-David, N. Nisan, and B. Pinkas, “Fairplaymp: a system for secure multi-party computation,” in *Proceedings of the 15th ACM conference on Computer and communications security*, 2008, pp. 257–266. [Cited on pages 26 and 27.]

[81] D. Demmler, T. Schneider, and M. Zohner, “Aby-a framework for efficient mixed-protocol secure two-party computation.” in *NDSS*, 2015. [Cited on pages 26 and 27.]

[82] ——, “Aby-a framework for efficient mixed-protocol secure two-party computation.” in *NDSS*, 2015. [Cited on pages 26, 27, 53, and 54.]

[83] T. K. Frederiksen, T. P. Jakobsen, J. B. Nielsen, and R. Trifiletti, “Tinylego: An interactive garbling scheme for maliciously secure two-party computation.” *IACR Cryptol. ePrint Arch.*, vol. 2015, p. 309, 2015. [Cited on page 27.]

[84] R. Trifiletti, “Tinylego,” available at: https://github.com/AarhusCrypto/TinyLEGO (Accessed 11 December 2020), 2016. [Cited on page 27.]

[85] ——, “Duplo,” available at: https://github.com/AarhusCrypto/DUPLO (Accessed 11 December 2020), 2016. [Cited on page 27.]

[86] V. Kolesnikov, J. B. Nielsen, M. Rosulek, N. Trieu, and R. Trifiletti, “Duplo: unifying cut-and-choose for garbled circuits,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 3–20. [Cited on page 27.]

[87] Y. Lindell, B. Pinkas, N. P. Smart, and A. Yanai, "Efficient constant round multi-party computation combining bmr and spdz," in *Annual Cryptology Conference*. Springer, 2015, pp. 319–338. [Cited on page 27.]

[88] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart, "Practical covertly secure mpc for dishonest majority–or: breaking the spdz limits," in *European Symposium on Research in Computer Security*. Springer, 2013, pp. 1–18. [Cited on page 27.]

[89] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979. [Cited on page 28.]

[90] K. V. Jónsson, G. Kreitz, and M. Uddin, "Secure multi-party sorting and applications." *IACR Cryptol. ePrint Arch.*, vol. 2011, p. 122, 2011. [Cited on page 28.]

[91] Y. Huang, D. Evans, J. Katz, and L. Malka, "Faster secure two-party computation using garbled circuits." in *USENIX Security Symposium*, vol. 201, no. 1, 2011, pp. 331–335. [Cited on page 28.]

[92] J. S. Resende, P. R. Sousa, R. Martins, and L. Antunes, "Breaking mpc implementations through compression," *International Journal of Information Security*, vol. 18, no. 4, pp. 505–518, 2019. [Cited on pages 28 and 57.]

[93] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 19–38. [Cited on page 28.]

[94] M. Andrychowicz, S. Dziembowski, D. Malinowski, and Ł. Mazurek, "Secure multiparty computations on bitcoin, 2014," in *35th IEEE Symposium on Security and Privacy (Oakland). ACM*, 2014. [Cited on page 29.]

[95] B. Yuan, W. Lin, and C. McDonnell, "Blockchains and electronic health records," *Mcdonnell. mit. edu*, 2016. [Cited on page 31.]

[96] M. Herlihy and M. Moir, "Enhancing accountability and trust in distributed ledgers," *arXiv preprint arXiv:1606.07490*, 2016. [Cited on page 31.]

[97] M. Trnka and T. Cerny, "Identity management of devices in internet of things environment," in *2016 6th international conference on it convergence and security (ICITCS)*. IEEE, 2016, pp. 1–4. [Cited on page 32.]

[98] T. El Maliki and J.-M. Seigneur, "A survey of user-centric identity management technologies," in *The International Conference on Emerging Security Information, Systems, and Technologies (SECUREWARE 2007)*. IEEE, 2007, pp. 12–17. [Cited on pages 32 and 33.]

[99] J. Brainard, A. Juels, R. L. Rivest, M. Szydlo, and M. Yung, "Fourth-factor authentication: somebody you know," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 168–178. [Cited on page 32.]

[100] P. Angin, B. Bhargava, R. Ranchal, N. Singh, M. Linderman, L. B. Othmane, and L. Lilien, "An entity-centric approach for privacy and identity management in cloud computing," in *2010 29th IEEE symposium on reliable distributed systems*. IEEE, 2010, pp. 177–183. [Cited on page 33.]

[101] Y. Cao and L. Yang, "A survey of identity management technology," in *2010 IEEE International Conference on Information Theory and Information Security*. IEEE, 2010, pp. 287–293. [Cited on page 33.]

[102] D. W. Chadwick, "Federated identity management," in *Foundations of security analysis and design V*. Springer, 2009, pp. 96–120. [Cited on page 33.]

[103] "Security assertion markup language (saml) v2.0 technical overview. working draft 10, 9 october 2006." available at: https://bit.ly/2fbPWjM, 2006, (Accessed 27 January 2017). [Cited on page 33.]

[104] D. Hardt *et al.*, "The oauth 2.0 authorization framework," RFC 6749, October, Tech. Rep., 2012. [Cited on page 33.]

[105] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore, "Openid connect core 1.0 incorporating errata set 1," *The OpenID Foundation, specification*, 2014. [Cited on page 33.]

[106] C. Chibelushi, A. Eardley, and A. Arabo, "Identity management in the internet of things: the role of manets for healthcare applications," *Computer Science and Information Technology*, vol. 1, no. 2, pp. 73–81, 2013. [Cited on page 33.]

[107] V. Prevelakis and D. Spinellis, "Sandboxing applications." in *USENIX Annual Technical Conference, FREENIX Track*, 2001, pp. 119–126. [Cited on page 33.]

[108] "What's shibboleth?" available at: https://shibboleth.net/index/, (Accessed 2 February 2019). [Cited on pages 34 and 101.]

[109] R. L. Morgan, S. Cantor, S. Carmody, W. Hoehn, and K. Klingenstein, "Federated security: The shibboleth approach," *Educause Quarterly*, vol. 27, no. 4, pp. 12–17, 2004. [Cited on pages 34, 101, and 103.]

[110] P. Mahalle, S. Babar, N. R. Prasad, and R. Prasad, "Identity management framework towards internet of things (iot): Roadmap and key challenges," in *International Conference on Network Security and Applications*. Springer, 2010, pp. 430–439. [Cited on pages 34 and 35.]

[111] R. Roman, P. Najera, and J. Lopez, "Securing the internet of things," *Computer*, no. 9, pp. 51–58, 2011. [Cited on pages 34 and 111.]

[112] "Open source identity and access management - for modern applications and services," available at: http://www.keycloak.org/ (Accessed 19 July 2017). [Cited on page 34.]

[113] P. PRIME, "Identity management for europe," 2008. [Cited on page 34.]

[114] 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*. ACM, 2002, pp. 21–30. [Cited on pages 34 and 36.]

[115] I. Alqassem and D. Svetinovic, "A taxonomy of security and privacy requirements for the internet of things (iot)," in *2014 IEEE International Conference on Industrial Engineering and Engineering Management*. IEEE, 2014, pp. 1244–1248. [Cited on page 34.]

[116] K.-Y. Lam and C.-H. Chi, "Identity in the internet-of-things (iot): New challenges and opportunities," in *International Conference on Information and Communications Security*. Springer, 2016, pp. 18–26. [Cited on pages 34 and 35.]

[117] S. Devadas, E. Suh, S. Paral, R. Sowell, T. Ziola, and V. Khandelwal, "Design and implementation of puf-based" unclonable" rfid ics for anti-counterfeiting and security applications," in *2008 IEEE international conference on RFID*. IEEE, 2008, pp. 58–64. [Cited on page 35.]

[118] P. Tuyls and L. Batina, "Rfid-tags for anti-counterfeiting," in *Cryptographers' Track at the RSA Conference*. Springer, 2006, pp. 115–131.

[119] A. Van Herrewege, S. Katzenbeisser, R. Maes, R. Peeters, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann, "Reverse fuzzy extractors: Enabling lightweight mutual authentication for puf-enabled rfids," in *International Conference on Financial Cryptography and Data Security*. Springer, 2012, pp. 374–389. [Cited on page 35.]

[120] C. Helfmeier, C. Boit, D. Nedospasov, S. Tajik, and J.-P. Seifert, "Physical vulnerabilities of physically unclonable functions," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–4. [Cited on page 35.]

[121] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 237–249.

[122] Y. Ikezaki, Y. Nozaki, and M. Yoshikawa, "Deep learning attack for physical unclonable function," in *2016 IEEE 5th Global Conference on Consumer Electronics*. IEEE, 2016, pp. 1–2.

[123] F. Ganji, S. Tajik, and J.-P. Seifert, "A fourier analysis based attack against physically unclonable functions," in *International Conference on Financial Cryptography and Data Security*. Springer, 2018, pp. 310–328. [Cited on page 35.]

[124] P. Butkus *et al.*, "A user centric identity management for internet of things," in *2014 International Conference on IT Convergence and Security (ICITCS)*. IEEE, 2014, pp. 1–4. [Cited on page 35.]

[125] C. De Terwangne, "Article 5. principles relating to processing of personal data," *The EU General Data Protection Regulation (GDPR) A Commentary*, 2020. [Cited on page 35.]

[126] A. C. Sarma and J. Girão, "Identities in the future internet of things," *Wireless personal communications*, vol. 49, no. 3, pp. 353–363, 2009. [Cited on page 35.]

[127] J. M. Such, A. Espinosa, A. Garcia-Fornes, and V. Botti, "Partial identities as a foundation for trust and reputation," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 7, pp. 1128–1136, 2011. [Cited on page 35.]

[128] J. B. Bernabé, J. L. H. Ramos, and A. F. Gómez-Skarmeta, "Holistic privacy-preserving identity management system for the internet of things." *Mobile Information Systems*, vol. 2017, pp. 6 384 186–1, 2017. [Cited on pages 35, 36, 98, 101, and 103.]

[129] M. Chase, "Anonymous credentials: How to show credentials without compromising privacy," available at: https://bit.ly/2Fwi02b (Accessed 17 January 2019), 2011. [Cited on pages 35 and 99.]

[130] C. Paquin and G. Zaverucha, "U-prove cryptographic specification v1. 1," *Technical Report, Microsoft Corporation*, 2011. [Cited on page 36.]

[131] J. L. C. Sanchez, J. B. Bernabe, and A. F. Skarmeta, "Integration of anonymous credential systems in iot constrained environments," *IEEE Access*, vol. 6, pp. 4767–4778, 2018. [Cited on pages 36, 101, and 103.]

[132] "Identity management - keyrock," available at: https://bit.ly/2NdxhFG (Accessed 1 July 2018), 2018. [Cited on page 36.]

[133] "Sociotal identitymanager," available at: https://github.com/sociotal/IdentityManager (Accessed 2 February 2019). [Cited on page 36.]

[134] N. Sem-III, "Privacy based public key infrastructure (pki) using smart contract in blockchain technology," *2nd Advanced Workshop on Blockchain: Technology, Applications, Challenges*, 2017. [Cited on page 37.]

[135] Y. Cai and D. Zhu, "Fraud detections for online businesses: a perspective from blockchain technology," *Financial Innovation*, vol. 2, no. 1, p. 20, 2016. [Cited on page 37.]

[136] P. Saint-Andre, "How can blockchains improve the internet of things?" available at: https://coincenter.org/entry/how-can-blockchains-improve-the-internet-of-things (Accessed 1 June 2018), 2016. [Cited on page 37.]

[137] H. Dai, Z. Zheng, and Y. Zhang, "Blockchain for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8076–8094, 2019. [Cited on page 37.]

[138] S. Y. Lim, P. T. Fotsing, A. Almasri, O. Musa, M. L. M. Kiah, T. F. Ang, and R. Ismail, "Blockchain technology the identity management and authentication service disruptor: a survey," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 8, no. 4-2, pp. 1735–1745, 2018. [Cited on pages 37 and 40.]

[139] X. Zhu and Y. Badr, "Identity management systems for the internet of things: A survey towards blockchain solutions," *Sensors*, vol. 18, no. 12, p. 4215, 2018. [Cited on pages 38, 39, and 98.]

[140] A. Abraham, "Self-sovereign identity," *Styria. EGIZ. GV. AT*, 2017. [Cited on pages 38, 39, 101, and 103.]

[141] D. Khovratovich and J. Law, "Sovrin: digital identities in the blockchain era," *Github Commit by jasonalaw October*, vol. 17, 2017. [Cited on pages 38, 99, 100, 101, and 103.]

[142] P. Dunphy and F. A. Petitcolas, "A first look at identity management schemes on the blockchain," *IEEE Security & Privacy*, vol. 16, no. 4, pp. 20–29, 2018. [Cited on page 38.]

[143] A. Giaretta, S. Pepe, and N. Dragoni, "Uniquid: A quest to reconcile identity access management and the iot," in *International Conference on Objects, Components, Models and Patterns*. Springer, 2019, pp. 237–251. [Cited on pages 38, 39, and 101.]

[144] D. Shrier, W. Wu, and A. Pentland, "Blockchain & infrastructure (identity, data security)," *Massachusetts Institute of Technology-Connection Science*, vol. 1, no. 3, pp. 1–19, 2016. [Cited on page 38.]

[145] P.-L. Aublin, S. B. Mokhtar, and V. Quéma, "Rbft: Redundant byzantine fault tolerance," in *2013 IEEE 33rd International Conference on Distributed Computing Systems*. IEEE, 2013, pp. 297–306. [Cited on page 38.]

[146] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014. [Cited on page 38.]

[147] "Sovrin: A protocol and token for self-sovereign identity and decentralized trust. sovrin foundation." available at: https://bit.ly/2S2k5Fg (Accessed 1 June 2018), 2018. [Cited on pages 38 and 101.]

[148] S. Bertram and C.-P. Georg, "A privacy-preserving system for data ownership using blockchain and distributed databases," *arXiv preprint arXiv:1810.11655*, 2018. [Cited on page 39.]

[149] R. Dillet, "Shocard is a digital identity card on the blockchain," 2015. [Cited on page 39.]

[150] G. Caronni, "Walking the web of trust," in *Proceedings IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2000).* IEEE, 2000, pp. 153–158. [Cited on page 39.]

[151] A. Anantha, M. R. Krishnan, A. L. Marshall, K. R. Zargahi, and M. T. Abel, "Secure element authentication," 2016, uS Patent 9,509,686. [Cited on page 39.]

[152] A. Giaretta, S. Pepe, and N. Dragoni, "Uniquid: A quest to reconcile identity access management and the internet of things," *arXiv preprint arXiv:1905.04021*, 2019. [Cited on page 39.]

[153] K. T. Nguyen, M. Laurent, and N. Oualha, "Survey on secure communication protocols for the internet of things," *Ad Hoc Networks*, vol. 32, pp. 17–31, 2015. [Cited on page 40.]

[154] M. A. Ferrag, L. A. Maglaras, H. Janicke, J. Jiang, and L. Shu, "Authentication protocols for internet of things: a comprehensive survey," *Security and Communication Networks*, vol. 2017, 2017.

[155] A. Albalawi, A. Almrshed, A. Badhib, and S. Alshehri, "A survey on authentication techniques for the internet of things," in *2019 International Conference on Computer and Information Sciences (ICCIS).* IEEE, 2019, pp. 1–5.

[156] Y. Atwady and M. Hammoudeh, "A survey on authentication techniques for the internet of things," in *proceedings of the international conference on future networks and distributed systems*, 2017.

[157] M. Saadeh, A. Sleit, M. Qatawneh, and W. Almobaideen, "Authentication techniques for the internet of things: A survey," in *2016 cybersecurity and cyberforensics conference (CCC).* IEEE, 2016, pp. 28–34.

[158] M. El-hajj, M. Chamoun, A. Fadlallah, and A. Serhrouchni, "Taxonomy of authentication techniques in internet of things (iot)," in *2017 IEEE 15th Student Conference on Research and Development (SCOReD).* IEEE, 2017, pp. 67–71.

[159] J. Granjal, E. Monteiro, and J. S. Silva, "Security for the internet of things: a survey of existing protocols and open research issues," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1294–1312, 2015.

[160] M. T. Hammi, B. Hammi, P. Bellot, and A. Serhrouchni, “Bubbles of trust: A decentralized blockchain-based authentication system for iot,” *Computers & Security*, vol. 78, pp. 126–142, 2018.

[161] M. R. Balasubramaniam, R. Sathya, S. Ashicka, and S. SenthilKumar, “An analysis of rfid authentication schemes for internet of things (iot) in healthcare environment using elgamal elliptic curve cryptosystem,” *Int. J. Recent Trends Eng. Res.(IJRTER)*, vol. 2, no. 3, 2016.

[162] M. H. Afifi, L. Zhou, S. Chakrabartty, and J. Ren, “Dynamic authentication protocol using self-powered timers for passive internet of things,” *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2927–2935, 2017. [Cited on page 40.]

[163] S. Agrawal and P. Ahlawat, “A survey on the authentication techniques in internet of things,” in *2020 IEEE International Students’ Conference on Electrical, Electronics and Computer Science (SCEECS)*. IEEE, 2020, pp. 1–5. [Cited on page 40.]

[164] V. Shivraj, M. Rajan, M. Singh, and P. Balamuralidhar, “One time password authentication scheme based on elliptic curves for internet of things (iot),” in *2015 5th National Symposium on Information Technology: Towards New Smart World (NSITNSW)*. IEEE, 2015, pp. 1–6. [Cited on page 40.]

[165] N. Aboudagga, M. T. Refaei, M. Eltoweissy, L. A. DaSilva, and J.-J. Quisquater, “Authentication protocols for ad hoc networks: Taxonomy and research issues,” in *Proceedings of the 1st ACM International Workshop on Quality of Service &Amp; Security in Wireless and Mobile Networks*, ser. Q2SWinet ’05. New York, NY, USA: ACM, 2005, pp. 96–104. [Online]. Available: http://doi.acm.org/10.1145/1089761.1089777 [Cited on page 40.]

[166] “2018 global pki trends study,” available at: https://bit.ly/3780EW0 (Accessed 11 December 2020), 2018. [Cited on page 40.]

[167] E. Rescorla, *SSL and TLS: designing and building secure systems*. Addison-Wesley Reading, 2001, vol. 1. [Cited on page 40.]

[168] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman, “The secure real-time transport protocol (srtp),” Internet Requests for Comments, RFC Editor, RFC 3711, March 2004, available at: http://www.rfc-editor.org/rfc/rfc3711.txt (Accessed 16 June 2018). [Cited on page 41.]

[169] P. Zimmermann, A. Johnston, and J. Callas, “Zrtp: Media path key agreement for unicast secure rtp,” Internet Requests for Comments, RFC Editor, RFC 6189, April 2011, available at: http://www.rfc-editor.org/rfc/rfc6189.txt (Accessed 16 June 2018). [Cited on pages 41, 49, and 50.]

[170] J. Lancrenon, M. Škrobot, and Q. Tang, "Two more efficient variants of the j-pake protocol," in *International Conference on Applied Cryptography and Network Security*. Springer, 2016, pp. 58–76. [Cited on page 41.]

[171] F. Hao and P. Y. Ryan, "Password authenticated key exchange by juggling," in *International Workshop on Security Protocols*. Springer, 2008, pp. 159–171. [Cited on page 41.]

[172] F. Hao, "J-pake: Password-authenticated key exchange by juggling," 2017. [Cited on page 41.]

[173] ——, "Schnorr nizk proof: Non-interactive zero knowledge proof for discrete logarithm version 5," *Online]. Tersedia: https://tools. ietf. org/html/draft-hao-schnorr-05*, 2016. [Cited on page 41.]

[174] M. Toorani, "Security analysis of j-pake," in *Computers and Communication (ISCC), 2014 IEEE Symposium on*. Funchal, Portugal: IEEE, 2014, pp. 1–6. [Cited on page 41.]

[175] S. Martini, "Session key retrieval in j-pake implementations of openssl and openssh. (2010)," 2018, available at: http://seb.dbzteam.org/crypto/jpake-session-key-retrieval.pdf, (Accessed 16 June 2018). [Cited on page 41.]

[176] C. H. Daniel Kaiser, "Device pairing using short authentication strings," 2016, available at: https://tools.ietf.org/html/draft-ietf-dnssd-pairing-01 (Accessed 21 April 2017). [Cited on page 41.]

[177] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong, "Talking to strangers: Authentication in ad-hoc wireless networks," in *NDSS*. San Diego; CA; USA.: Network and Distributed System Security Symposium, 2002. [Cited on page 42.]

[178] A. Spahić, M. Kreutzer, M. Kähmer, and S. Chandratilleke, "Pre-authentication using infrared," in *Privacy, Security and Trust within the Context of Pervasive Computing*. Springer, 2005, pp. 105–112. [Cited on page 42.]

[179] S. Vaudenay, "Secure communications over insecure channels based on short authenticated strings," in *Proceedings of the 25th Annual International Conference on Advances in Cryptology*, ser. CRYPTO'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 309–326. [Online]. Available: http://dx.doi.org/10.1007/11535218_19 [Cited on page 42.]

[180] I. Buhan, J. Doumen, P. Hartel, and R. Veldhuis, *Feeling is Believing: a location limited channel based on grip pattern biometrics and cryptanalysis*, ser. CTIT technical reports series. Centrum voor Telematica en Informatie Technologie: Centrum voor Telematica en Informatie Technologie, 6 2006, no. 06-29, imported from CTIT. [Cited on page 42.]

[181] N. Asokan and P. Ginzboorg, "Key agreement in ad hoc networks," *Computer communications*, vol. 23, no. 17, pp. 1627–1637, 2000. [Cited on page 42.]

[182] D. Kelly and M. Hammoudeh, “Optimisation of the public key encryption infrastructure for the internet of things,” in *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*, 2018, pp. 1–5. [Cited on page 42.]

[183] V. Shivraj, M. Rajan, M. Singh, and P. Balamuralidhar, “One time password authentication scheme based on elliptic curves for internet of things (iot),” in *2015 5th National Symposium on Information Technology: Towards New Smart World (NSITNSW)*. IEEE, 2015, pp. 1–6. [Cited on page 42.]

[184] S. Rajagopalan, S. Janakiraman, A. Rengarajan, S. Rethinam, S. Arumugham, and G. Saravanan, “Iot framework for secure medical image transmission,” in *2018 international conference on computer communication and informatics (ICCCI)*. IEEE, 2018, pp. 1–5. [Cited on page 42.]

[185] M. N. Aman, K. C. Chua, and B. Sikdar, “Physically secure mutual authentication for iot,” in *2017 IEEE Conference on Dependable and Secure Computing*. IEEE, 2017, pp. 310–317. [Cited on page 42.]

[186] ——, “Mutual authentication in iot systems using physical unclonable functions,” *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1327–1340, 2017. [Cited on page 42.]

[187] M. Prince, “Technical details behind a 400gbps ntp amplification ddos attack,” *Cloudflare, Inc*, vol. 13, 2014. [Cited on page 43.]

[188] A. Rousskov, “Squid: Optimising web delivery,” 2013. [Online]. Available: http://www.squid-cache.org/ [Cited on page 43.]

[189] A. Rousskov and C. Tsantilas, “Squid-in-the-middle ssl bump,” 2019. [Online]. Available: https://wiki.squid-cache.org/Features/SslBump [Cited on page 43.]

[190] D. E. Eastlake, “Transport layer security (tls) extensions: Extension definitions,” *RFC*, vol. 6066, pp. 1–25, 2011. [Cited on page 43.]

[191] T. Dierks and E. Rescorla, “The transport layer security (tls) protocol version 1.2,” *RFC 5246, August*, 2008. [Cited on page 43.]

[192] B. Volz, “The dynamic host configuration protocol for ipv6 (dhcpv6) client fully qualified domain name (fqdn) option,” *RFC*, vol. 4704, pp. 1–15, 2006. [Cited on page 43.]

[193] N. A. Svee, “Filter tls traffic with iptables,” available at: https://github.com/Lochnair/xt_tls (Accessed 4 December 2019), 2016. [Cited on page 44.]

[194] W. M. Shbair, T. Cholez, A. Goichot, and I. Chrisment, “Efficiently bypassing sni-based https filtering,” in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 990–995. [Cited on page 44.]

[195] E. Rescorla, K. Oku, N. Sullivan, and C. A. Wood, “Encrypted server name indication for tls 1.3,” *IETF draft. Available at: https://tools. ietf. org/html/draft-ietf-tls-esni-02 (Accessed December 14th 2018)*, 2018. [Cited on page 44.]

[196] C. Haar and E. Buchmann, “Fane: A firewall appliance for the smart home,” in *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 2019, pp. 449–458. [Cited on page 44.]

[197] D. A. Sørensen, N. Vanggaard, and J. M. Pedersen, “Automatic profile-based firewall for iot devices,” *available at: https://bit.ly/3fjjCdA (Accessed 11 December 2020)*, 2017. [Cited on page 45.]

[198] E. Lastdrager, C. Hesselman, J. Jansen, and M. Davids, “Protecting home networks from insecure iot devices,” in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–6. [Cited on page 45.]

[199] M. Roesch, “Snort - lightweight intrusion detection for networks,” in *Proceedings of the 13th USENIX Conference on System Administration*, ser. LISA ’99. USA: USENIX Association, 1999, p. 229–238. [Cited on page 45.]

[200] A. M. Mandalari, R. Kolcun, H. Haddadi, D. J. Dubois, and D. Choffnes, “Towards automatic identification and blocking of non-critical iot traffic destinations,” *arXiv preprint arXiv:2003.07133*, 2020. [Cited on page 45.]

[201] Pihole, “Pi-hole® network-wide ad blocking,” 2015. [Online]. Available: https://pi-hole.net/ [Cited on page 45.]

[202] “Phishtank: An anti-phishing site,” *available at: https://www.phishtank.com (Accessed 11 December 2020)*, 2016. [Cited on page 45.]

[203] S. Chakraborty, C. Shen, K. R. Raghavan, Y. Shoukry, M. Millar, and M. Srivastava, “ipshield: a framework for enforcing context-aware privacy,” in *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, 2014, pp. 143–156. [Cited on page 45.]

[204] A. Das, M. Degeling, D. Smullen, and N. Sadeh, “Personalized privacy assistants for the internet of things: providing users with notice and choice,” *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 35–46, 2018. [Cited on page 45.]

[205] O. Drozd and S. Kirrane, “I agree: Customize your personal data processing with the core user interface,” in *International Conference on Trust and Privacy in Digital Business*. Springer, 2019, pp. 17–32. [Cited on page 45.]

[206] A. Al-Hasnawi and L. Lilien, “Pushing data privacy control to the edge in iot using policy enforcement fog module,” in *Companion Proceedings of the10th International Conference on Utility and Cloud Computing*, 2017, pp. 145–150. [Cited on page 46.]

[207] A. Das, M. Degeling, D. Smullen, and N. Sadeh, "Personal privacy assistants for the internet of things." [Cited on page 46.]

[208] V. Sivaraman, H. H. Gharakheili, A. Vishwanath, R. Boreli, and O. Mehani, "Network-level security and privacy control for smart-home iot devices," in *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2015, pp. 163–167. [Cited on page 46.]

[209] C. Adams and S. Lloyd, *Understanding PKI: Concepts, Standards, and Deployment Considerations*, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002. [Cited on page 49.]

[210] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing robust and ubiquitous security support for mobile ad hoc networks." in *icnp*, vol. 1. Citeseer, 2001, pp. 251–260. [Cited on page 49.]

[211] P. R. Sousa, J. S. Resende, R. Martins, and L. Antunes, "Internet of things security with multi-party computation (mpc)," Nov 2019. [Cited on page 49.]

[212] P. R. Sousa, A. Cirne, J. S. Resende, R. Martins, and L. Antunes, "ptasc: trustable autonomous secure communications," in *Proceedings of the 20th International Conference on Distributed Computing and Networking*, 2019, pp. 193–202. [Cited on page 49.]

[213] D. H. Seo and P. Sweeney, "Simple authenticated key agreement algorithm," *Electronics Letters*, vol. 35, no. 13, pp. 1073–1074, 1999. [Cited on page 49.]

[214] P. Zimmermann, A. Johnston, and J. Callas, "Zrtp: Media path key agreement for unicast secure rtp," 2011. [Cited on pages 51, 52, and 63.]

[215] W. Dittmann, "Zrtpcpp - c++ implementation of zrtp protocol," 2018, available at: https://github.com/wernerd/ZRTPCPP (Accessed 16 June 2018). [Cited on page 53.]

[216] D. Demmler, T. Schneider, and M. Zohner, "Aby-a framework for efficient mixed-protocol secure two-party computation." in *NDSS*, 2015. [Cited on page 54.]

[217] R. Developers, "Raspbian," 2018, available at: https://www.raspbian.org/ (Accessed 16 June 2018). [Cited on page 58.]

[218] "pigpio is a c library for the raspberry which allows control of the general purpose input outputs (gpio)." available at: https://github.com/joan2937/pigpio (Accessed 4 December 2019), 2012. [Cited on page 58.]

[219] J. Postel, "Transmission control protocol," Internet Requests for Comments, RFC Editor, STD 7, September 1981, available at: http://www.rfc-editor.org/rfc/rfc793.txt (Accessed 11 December 2020). [Cited on page 58.]

[220] K. Isoyama, “Multiplex server system and server multiplexing method,” Sep. 20 2007, uS Patent App. 11/723,499. [Cited on page 58.]

[221] E. S. Bender and C.-K. K. Yee, “Method and apparatus for managing thread execution in a multithread application,” May 8 2007, uS Patent 7,216,346. [Cited on page 58.]

[222] I. Free Software Foundation, “Gnu [u]common c++,” 2018, available at: https://www.gnu.org/software/commoncpp/ (Accessed 16 June 2018). [Cited on page 59.]

[223] D. Sisalem, J. Floroiu, J. Kuthan, U. Abend, and H. Schulzrinne, *SIP security*. John Wiley & Sons, 2009. [Cited on page 63.]

[224] J. S. Resende, P. R. Sousa, and L. Antunes, “Evaluating the privacy properties of secure voip metadata,” in *International Conference on Trust and Privacy in Digital Business*. Springer, 2018, pp. 57–68. [Cited on page 63.]

[225] B. Brenner, “Let’s encrypt issues certs to ’paypal’ phishing sites: how to protect yourself (2017)),” 2017, available at: http://bit.ly/2i7Z4bT (Accessed 19 May 2017). [Cited on page 64.]

[226] M. Petraschek, T. Hoeher, O. Jung, H. Hlavacs, and W. N. Gansterer, “Security and usability aspects of man-in-the-middle attacks on zrtp.” *J. UCS*, vol. 14, no. 5, pp. 673–692, 2008. [Cited on page 65.]

[227] J. Vincent, “Lyrebird claims it can recreate any voice using just one minute of sample audio,” *The Verge*, vol. 24, 2017. [Cited on page 65.]

[228] R. Pass, “Bounded-concurrent secure multi-party computation with a dishonest majority,” in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, 2004, pp. 232–241. [Cited on page 65.]

[229] Y. H. Hwang, “Iot security & privacy: threats and challenges,” in *Proceedings of the 1st ACM workshop on IoT privacy, trust, and security*, 2015, pp. 1–1. [Cited on page 67.]

[230] S. Gueron and V. Krasnov, “Fast prime field elliptic-curve cryptography with 256-bit primes,” *Journal of Cryptographic Engineering*, vol. 5, no. 2, pp. 141–151, 2015. [Cited on page 68.]

[231] E. Barker, D. Johnson, and M. Smid, “Nist special publication 800-56a: Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography (revised),” *Comput. Secur. Natl. Inst. Stand. Technol.(NIST), Publ. by NIST*, 2007. [Cited on page 69.]

[232] I. F. Blake, G. Seroussi, and N. P. Smart, *Advances in elliptic curve cryptography*. Cambridge University Press, 2005, vol. 317. [Cited on page 70.]

[233] S. Cheshire, “Zero configuration networking (zeroconf),” *http://www.zeroconf.org*, 2007. [Cited on page 70.]

[234] J. F. Dray, "Nist special publication 800-73 interfaces for personal identity," Ph.D. dissertation, National Institute of Standards and Technology, 2005. [Cited on page 73.]

[235] C. Kolbel, "privacyidea authentication system, release 2.17," *online],[retrieved Jan. 23, 2017]. Retrieved from the internet*, 2017. [Cited on page 74.]

[236] H. Brekalo, R. Strackx, and F. Piessens, "Mitigating password database breaches with intel sgx," in *Proceedings of the 1st Workshop on System Software for Trusted Execution*, 2016, pp. 1–6. [Cited on page 76.]

[237] N. Popp, "Token authentication system and method," Jan. 28 2014, uS Patent 8,639,628. [Cited on page 77.]

[238] K. J. Singh and D. S. Kapoor, "Create your own internet of things: A survey of iot platforms." *IEEE Consumer Electronics Magazine*, vol. 6, no. 2, pp. 57–68, 2017. [Cited on page 77.]

[239] T. Ammari, J. Kaye, J. Y. Tsai, and F. Bentley, "Music, search, and iot: How people (really) use voice assistants," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 26, no. 3, pp. 1–28, 2019. [Cited on page 79.]

[240] J. Lau, B. Zimmerman, and F. Schaub, "Alexa, are you listening? privacy perceptions, concerns and privacy-seeking behaviors with smart speakers," *Proceedings of the ACM on Human-Computer Interaction*, vol. 2, no. CSCW, pp. 1–31, 2018. [Cited on pages 79 and 80.]

[241] A. Purington, J. G. Taft, S. Sannon, N. N. Bazarova, and S. H. Taylor, "" alexa is my new bff" social roles, user satisfaction, and personification of the amazon echo," in *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, 2017, pp. 2853–2859. [Cited on page 79.]

[242] K. Noda, "Google home: smart speaker as environmental control unit," *Disability and rehabilitation: assistive technology*, vol. 13, no. 7, pp. 674–675, 2018. [Cited on page 79.]

[243] C. Foxx, "Apple reveals homepod smart speaker," *BBC News, Jun*, vol. 5, p. 6, 2017. [Cited on page 79.]

[244] A. A. Berger, "The amazon echo," in *Perspectives on Everyday Life*. Springer, 2018, pp. 79–82. [Cited on page 79.]

[245] J. Kastrenakes, "Google sold over 6 million home speakers since mid-october." available at: https://bit.ly/3qOww9S (Accessed 11 December 2020), 2018. [Cited on page 79.]

[246] N. Gartner, "Gartner says worldwide spending on vpa-enabled wireless speakers will top $2 billion by 2020," *Gartner Newsroom*, 2016. [Cited on page 80.]

[247] J. Y. Tsai, P. Kelley, P. Drielsma, L. F. Cranor, J. Hong, and N. Sadeh, "Who's viewed you? the impact of feedback in a mobile location-sharing application," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2009, pp. 2003–2012. [Cited on page 80.]

[248] P. Regulation, "Regulation (eu) 2016/679 of the european parliament and of the council," *REGULATION (EU)*, vol. 679, p. 2016, 2016. [Cited on page 84.]

[249] Efergy, "Efergy home energy monitors: Electricity usage power monitor," available at: https://efergy.com/about-efergy/ (Accessed 11 December 2020), 2006. [Cited on page 87.]

[250] E. de Matos, L. A. Amaral, R. T. Tiburski, M. C. Schenfeld, D. F. de Azevedo, and F. Hessel, "A sensing-as-a-service context-aware system for internet of things environments," in *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2017, pp. 724–727. [Cited on page 87.]

[251] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 414–454, 2013. [Cited on page 88.]

[252] S. Sukode, S. Gite, and H. Agrawal, "Context aware framework in iot: a survey," *International Journal*, vol. 4, no. 1, 2015.

[253] O. B. Sezer, E. Dogdu, and A. M. Ozbayoglu, "Context-aware computing, learning, and big data in internet of things: a survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 1–27, 2017. [Cited on page 88.]

[254] E. de Matos, R. T. Tiburski, L. A. Amaral, and F. Hessel, "Providing context-aware security for iot environments through context sharing feature," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2018, pp. 1711–1715. [Cited on page 88.]

[255] X. Zhao, K. K. Sajan, G. S. Ramachandran, and B. Krishnamachari, "Demo abstract: The intelligent iot integrator data marketplace-version 1," in *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2020, pp. 270–271. [Cited on page 88.]

[256] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "Blockchain for iot security and privacy: The case study of a smart home," in *2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops)*. IEEE, 2017, pp. 618–623. [Cited on page 97.]

[257] M. Banerjee, J. Lee, and K.-K. R. Choo, "A blockchain future for internet of things security: A position paper," *Digital Communications and Networks*, vol. 4, no. 3, pp. 149–160, 2018. [Cited on page 97.]

[258] M. A. Walker, A. Dubey, A. Laszka, and D. C. Schmidt, "Platibart: a platform for transactive iot blockchain applications with repeatable testing," in *Proceedings of the 4th Workshop on Middleware and Applications for the Internet of Things*. ACM, 2017, pp. 17–22.

[259] E. F. Jesus, V. R. Chicarino, C. V. de Albuquerque, and A. A. d. A. Rocha, "A survey of how to use blockchain to secure internet of things and the stalker attack," *Security and Communication Networks*, vol. 2018, 2018. [Cited on page 97.]

[260] M. A. Akkaş, R. Sokullu, and H. E. Çetin, "Healthcare and patient monitoring using iot," *Internet of Things*, vol. 11, p. 100173, 2020. [Cited on page 98.]

[261] Y. Yang, X. Ding, H. Lu, J. Weng, and J. Zhou, "Self-blindable credential: towards anonymous entity authentication upon resource constrained devices," in *Information Security*. Springer, 2015, pp. 238–247. [Cited on page 99.]

[262] J. Garcia-Alfaro, G. Navarro-Arribas, H. Hartenstein, and J. Herrera-Joancomartí, "Data privacy management, cryptocurrencies and blockchain technology," 2017. [Cited on page 100.]

[263] M. Layouni and H. Vangheluwe, "Anonymous k-show credentials," in *European Public Key Infrastructure Workshop*. Springer, 2007, pp. 181–192. [Cited on page 100.]

[264] J. Lapon, M. Kohlweiss, B. De Decker, and V. Naessens, "Analysis of revocation strategies for anonymous idemix credentials," in *IFIP International Conference on Communications and Multimedia Security*. Springer, 2011, pp. 3–17. [Cited on page 100.]

[265] C. Chung, K. Lee, J. Yun, and D. Won, "An improved anonymous electronic prescription scheme," in *International Conference on Future Generation Information Technology*. Springer, 2011, pp. 293–300. [Cited on page 100.]

[266] J. R. Prins and B. U. Cybercrime, "Diginotar certificate authority breach'operation black tulip'," *Fox-IT, November*, 2011. [Cited on page 100.]

[267] K. Zetter, "Diginotar files for bankruptcy in wake of devastating hack," *Wired magazine, September*, 2011. [Cited on page 100.]

[268] D. Fisher, "Diginotar says its ca infrastructure was compromised," available at: https://bit.ly/377LuzT (Accessed 9 December 2020), 2011. [Cited on page 100.]

[269] "How blockchain startups will solve the identity crisis for the internet of things." available at: https://bit.ly/2ALLp48 (Accessed 10 Dec 2020), 2017. [Cited on page 100.]

[270] D. Recordon and D. Reed, “Openid 2.0: a platform for user-centric identity management,” in *Proceedings of the second ACM workshop on Digital identity management*. ACM, 2006, pp. 11–16. [Cited on pages 101 and 103.]

[271] “Openid organization,” available at: https://github.com/openid (Accessed 4 May 2018), 2017. [Cited on pages 101 and 103.]

[272] “Sociotal,” available at: https://github.com/sociotal/SOCIOTAL (Accessed 4 May 2018), 2015. [Cited on page 103.]

[273] “uport contracts for managing identity,” available at: https://github.com/uport-project/uport-identity (Accessed 4 May 2018), 2017. [Cited on page 103.]

[274] “Hyperledger indy,” available at: https://github.com/hyperledger/indy-node, 2016, (Accessed 4 May 2018). [Cited on page 103.]

[275] “Consensys, identity for blockchain vs blockchain for identity,” available at: https://bit.ly/2FWPerQ, 2016, (Accessed 2 February 2018). [Cited on page 104.]

[276] “Apple watch saves the life of florida teen with a life-threatening disease.” available at: https://bit.ly/2MXG672 (Accessed 28 June 2018), 2018. [Cited on page 105.]

[277] I. Chiuchisan, H.-N. Costin, and O. Geman, “Adopting the internet of things technologies in health care systems,” in *2014 International Conference and Exposition on Electrical and Power Engineering (EPE)*. IEEE, 2014, pp. 532–535. [Cited on page 106.]

[278] “Intensive care unit,” available at: https://bit.ly/2tHGzBE, 2017, (Accessed 29 June 2018). [Cited on page 106.]

[279] I. Chiuchisan, H.-N. Costin, and O. Geman, “Adopting the internet of things technologies in health care systems,” in *Electrical and Power Engineering (EPE), 2014 International Conference and Exposition on*. IEEE, 2014, pp. 532–535. [Cited on page 106.]

[280] I. Azimi, A. Anzanpour, A. M. Rahmani, P. Liljeberg, and H. Tenhunen, “Self-aware early warning score system for iot-based personalized healthcare,” in *eHealth 360*. Springer, 2017, pp. 49–55. [Cited on page 107.]

[281] F. Fernandez and G. C. Pallis, “Opportunities and challenges of the internet of things for healthcare: Systems engineering perspective,” in *2014 4th International Conference on Wireless Mobile Communication and Healthcare-Transforming Healthcare Through Innovations in Mobile and Wireless Technologies (MOBIHEALTH)*. IEEE, 2014, pp. 263–266. [Cited on page 107.]

[282] “How to draw for kids,” available at: https://bit.ly/2NcrsbR, 2017, (Accessed 29 June 2018). [Cited on page 108.]

[283] A. Pressman, "Fitbit strikes deal with google that could lead to wearables collaboration." available at: https://bit.ly/2WPBCUC (Accessed 26 June 2018), 2018. [Cited on page 108.]

[284] A. L. G. de Sousa Brandão, "Systems hardening through the use of secure enclaves," 2020. [Cited on page 117.]

[285] T. Matrix, "Twisted matrix labs," *Online. Accessed*, vol. 8, 2015.

[286] A. Aly, "Network flow problems with secure multiparty computation." Ph.D. dissertation, Catholic University of Louvain, Louvain-la-Neuve, Belgium, 2015.

[287] A. Martelli, "General multi-precision arithmetic for python 2.6+/3+ (gmp, mpir, mpfr, mpc)," available at: https://github.com/aleaxit/gmpy (Accessed 11 December 2020), 2015.

[288] N. M. Edwin, "Software frameworks, architectural and design patterns," *Journal of Software Engineering and Applications*, vol. 2014, 2014.

[289] S. Rass, P. Schartner, and M. Brodbeck, "Private function evaluation by local two-party computation," *EURASIP Journal on Information Security*, vol. 2015, no. 1, p. 7, 2015.

[290] Y. Ejgenberg, M. Farbstein, M. Levy, and Y. Lindell, "Scapi: The secure computation application programming interface." *IACR Cryptol. ePrint Arch.*, vol. 2012, p. 629, 2012.

[291] I. P. Vuksanovic and B. Sudarevic, "Use of web application frameworks in the development of small applications," in *2011 Proceedings of the 34th International Convention MIPRO*. IEEE, 2011, pp. 458–462.

[292] W. Henecka, S. K ögl, A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, "Tasty: tool for automating secure two-party computations," in *Proceedings of the 17th ACM conference on Computer and communications security*, 2010, pp. 451–462.

[293] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos, "Sepia: Security through private information aggregation," *arXiv preprint arXiv:0903.4258*, 2009.

[294] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos, "Sepia: Privacy-preserving aggregation of multi-domain network events and statistics," *Network*, vol. 1, no. 101101, 2010.

[295] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos, "Sepia: security through private information aggregation (2009)," *arXiv preprint arXiv:0903.4258*, 2009.

[296] C. Dannen, *Introducing Ethereum and solidity*. Springer, 2017, vol. 1.

[297] N. M. Edwin, "Software frameworks, architectural and design patterns," *Journal of Software Engineering and Applications*, vol. 2014, 2014.

[298] A. Ackerman, A. Chang, N. Diakun-Thibault, L. Forni, F. Landa, J. Mayo, and R. van Riezen, "Blockchain and health it: Algorithms, privacy and data," *Project PharmOrchard of MIT's Experimental Learning "MIT FinTech: Future Commerce.", White Paper August*, 2016.

[299] S. Babar, P. Mahalle, A. Stango, N. Prasad, and R. Prasad, "Proposed security model and threat taxonomy for the internet of things (iot)," in *International Conference on Network Security and Applications*. Springer, 2010, pp. 420–429.

[300] "Web services federation language (ws-federation) version 1.0, oasis," available at: https://ibm.co/2WNfFFG, 2003, (Accessed 27 January 2017).

[301] "Web services trust language (ws-trust)", microsoft, ibm, opennetwork, layer 7, computer associates, verisign, bea, oblix, reactivity, rsa security, ping identity, verisign, actional," available at: https://bit.ly/2t8TprU, 2005, (Accessed 2 February 2019).

[302] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. bft replication," in *International workshop on open problems in network security*. Springer, 2015, pp. 112–125.

[303] D. Schürmann, F. Kabus, G. Hildermeier, and L. Wolf, "CVE-2016-6271." Available from MITRE, CVE-ID CVE-2016-6271., Feb. 2016, available at: https://nvd.nist.gov/vuln/detail/CVE-2016-6271 (Accessed 16 June 2018).

[304] SilentCircle, "Zrtpcpp - c++ implementation of zrtp protocol," 2018, available at: https://github.com/SilentCircle/ZRTPCPP (Accessed 16 June 2018).

[305] I. Free Software Foundation, "Gnu ccrtp," 2018, available at: https://www.gnu.org/software/ccrtp (Accessed 16 June 2018.

[306] J. Bosch, "Design patterns as language constructs," *Journal of Object-Oriented Programming*, vol. 11, pp. 18–32, 1998.

[307] jitsi, "zrtp4j," 2018, available at: https://github.com/jitsi/zrtp4j (Accessed 16 June 2018).

[308] P. Thermos and A. Takanen, *Securing VoIP Networks: Threats, Vulnerabilities, and Countermeasures*. Addison-Wesley Professional, 2007.

[309] B. C. Neuman and T. Ts'o, "Kerberos: An authentication service for computer networks," *IEEE Communications magazine*, vol. 32, no. 9, pp. 33–38, 1994.

[310] P. R. Sousa, L. Antunes, and R. Martins, "The present and future of privacy-preserving computation in fog computing," in *Fog Computing in the Internet of Things*. Springer, 2018, pp. 51–69.

[311] P. R. Sousa, R. Martins, and L. Antunes, "Empowering users through a privacy middleware watchdog," in *Trust, Privacy and Security in Digital Business*, S. Gritzalis, E. R. Weippl, G. Kotsis, A. M. Tjoa, and I. Khalil, Eds. Cham: Springer International Publishing, 2020, pp. 156–170.

[312] P. R. Sousa, J. S. Resende, R. Martins, and L. Antunes, "The case for blockchain in iot identity management," *Journal of Enterprise Information Management*, 2020.

[313] P. R. Sousa, J. S. Resende, R. Martins, and L. Antunes, "Secure provisioning for achieving end-to-end secure communications," in *International Conference on Ad-Hoc Networks and Wireless*. Springer, 2019, pp. 498–507.

[314] C. Perera, C. H. Liu, and S. Jayawardena, "The emerging internet of things marketplace from an industrial perspective: A survey," *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 4, pp. 585–598, 2015.

[315] A. Bahga and V. Madisetti, *Internet of Things: A hands-on approach*. Vpt, 2014.

[316] A. B. Brush, E. Filippov, D. Huang, J. Jung, R. Mahajan, F. Martinez, K. Mazhar, A. Phanishayee, A. Samuel, J. Scott *et al.*, "Lab of things: a platform for conducting studies with connected devices in multiple homes," in *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, 2013, pp. 35–38.

[317] C. Perera, R. Ranjan, L. Wang, S. U. Khan, and A. Y. Zomaya, "Big data privacy in the internet of things era," *IT Professional*, vol. 17, no. 3, pp. 32–39, 2015.

[318] M. Popescu, L. Baruh, M. Popescu, L. Baruh, P. Messaris, and L. Humphreys, "Consumer surveillance and distributive privacy harms in the age of big data," *Digital media: Transformations in human communication*, pp. 313–327, 2017.