Empowering Users Through a Privacy Middleware Watchdog

Patrícia R. Sousa[0000−0002−0268−9134], Rolando Martins[0000−0002−1838−1417], and Luís Antunes[0000−0002−9988−594X]

CRACS / INESC TEC
{psousa,rmartins,lfa}@dcc.fc.up.pt

Abstract. The ever-increasing number of interconnected devices in smart environments, i.e., homes and cities, is bolstering the amount of data generated and exchanged. These devices can range from small embedded platforms, such as those included in home appliances, to critical operational systems, such as traffic lights. However, this increasing adoption is raising significant security and privacy concerns. Although some researchers have already solved some of these issues, data privacy still lacks a viable solution, especially when considering a flexible, decentralized approach to avoid a central overseer. One of the biggest challenges regarding privacy is the lack of transparency about how data flows are mediated and regulated as, often, these resources share data with external entities without the users' knowledge. We argue that a novel data-sharing control mechanism is required to properly control users' privacy and their respective Internet of Things (IoT) devices. This work focuses on a middleware layer solution for the IoT devices, which allows the control of the data generated by the device by its owner. The platform places the user as an active participant in the data market, behaving as its own data intermediary for potential consumers by monitoring, controlling, and negotiating the usage of their data.

The final authenticated version is available online at https://doi.org/10.1007/978-3-030-58986-8_11.

1 Introduction

The IoT is growing continuously and, according to *National Public Radio & Edison Research* [25], 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. People are increasingly adopting the concept of smart homes by acquiring more smart devices, such as smart lamps or thermostats, helping to save money. Authenticated users can remotely control many devices via their smartphone, such as refrigerators or garage doors. Some examples include alerts from refrigerators when items are missing or a pot that can be programmed to irrigate plants. Along with the usability enhancements, it is also possible to make homes more secure with wireless security cameras, sensors, and

smoke alarms. Besides, this 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.

However, what is the cost of having all smart devices connected to the Internet, regarding some fundamental human rights, such as 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 send it as telemetry to their manufactures.

Jingjing Ren et al. [24] conducted a study with 81 different smart devices to examine data sharing activities. The authors found that 72 of the 81 IoT devices share sensitive data, such as IP addresses, device specifications and settings, usage habits, and location data, with third parties that were utterly unrelated with the original manufacturer, such as advertisers. This type of sharing can violate the user's privacy because, for example, according to the GDPR, 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 [12]. This lack of consumer awareness is rapidly raising concerns about data privacy [17] 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.

Recent literature [4, 36, 37] highlighted that privacy concerns could be a significant barrier to the growth of IoT, highlighting security and privacy as a significant IoT research challenge. Some of the identified missing features include: a) decentralized authentication and trust models; b) data protection technologies; c) data ownership; d) repository data management; e) access and use rights; f) integration or connection with privacy preservation structures; and g) management of privacy policies. Currently, managing identity and access control of "things" in an interconnected world remains an open issue.

This article 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.

2 Related Work

The development of middleware solutions for IoT is an active area of research. *Christoph Haar and Erik Buchmann* [14] introduced 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 the set of generated rules. *Daniel Amkær Sørensen et al.* [32] also propose a system that generates rules based on the real-time traffic analysis.

Pi-hole [22] 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* [20], is that the DNS server runs locally on the RP3, which inherently gives greater control and therefore privacy.

Security and Privacy for In-home Networks (SPIN) [16] 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* [26], or customized by the user.

More recently, *Olha Drozd and Sabrina Kirrane* [7] 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, for example. 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 purposes the UI for the idea. *Abduljaleel Al-Hasnawi and Leszek Lilien* [2] provide a solution that enforces privacy policies anywhere in the IoT, so it is more related to the data itself and the identification of 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.

Vijay Sivaraman et al. [31] propose an architecture with three components: Internet Service Provider (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 can be integrated, because 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.

3 Conceptual Design

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 has implemented the appropriate privacy and security mechanisms. However, this is not the case, mainly from the examples described in 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, for example, users can block marketing/advertising sites and only communicate with the manufacturer's domains. Note that 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. Finally, users will have to choose between usability and privacy.

On existing routers, users can change the set of rules and block specific traffic, without the need for a middleware. However, there are no configurable privacy platforms that show connections made by default by the device (configured by the manufacturer), and that allow blocking those connections "on the fly". The middleware allows users to monitor incoming and outgoing connections so that users can verify that the device is running an untrusted program and block or disable updates to specific resources.

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 with multiple routers (at home or work, for example), they can choose different permissions for their data depending on the device context, managing the data's life cycle. Users can also monetize data by selling them to external entities.

4 Architecture

The middleware consists of a secure data sharing model. From an architectural point of view, the system consists of 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 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 devices 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.

4.1 White-Box Devices

This type of device has two main components: a *Middleware* and a *Sensor API*.

Sensor 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 *Sensor API* to get access to the data. Then, the *Repository* receives the input from the

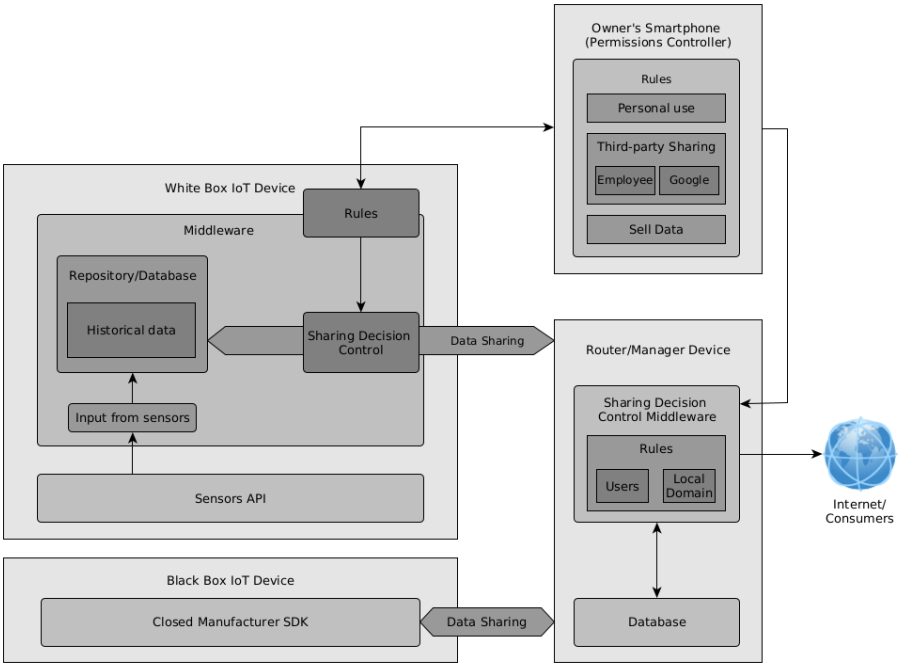

Fig. 1. Data Sharing Middleware

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.

4.2 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 has to perform all the traffic control (the router also has the middleware).

4.3 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 with entities unrelated to the manufacturer. In 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 Server Name Indication (SNI) [9]. According to the same use case, instead of showing only the *Google* entity with aggregated traffic, it shows detailed traffic in real-time (e.g. *192.168.1.6:38578 - 172.217.17.10:[443] 21:oauth2.googleapis.com*, *192.168.1.6:36366 - 216.58.201.234:[443] 32:embeddedassistant.googleapis.com*), and users can block or allow specific connections.

As mentioned in Section 3, 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 three main permissions to control the data sharing: *Personal Use*, *Third-party sharing* and *Sell Data*. In this case, both the IoT devices and the router implement all the rules defined on the smartphone.

Personal Use
Personal Use has two options: *yes* or *no*. 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.

Third-party Sharing
Third-party Sharing has two different components: *Internal Sharing* and *External Sharing*. *Internal Sharing* represents the data transmission that exists between the IoT device and the router to which it is connected. Owners can set different permissions according to the context of 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 on 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. *External Sharing* is explained in Section 4.3 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 the interests of a person or region. 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.

4.4 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 are synchronized with *yes*, these data can be transmitted to the *Database*, allowing data owners to query them offline. In the router, it is possible to make the interconnection between 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, regardless of the type of device. As we have black-box devices where we do not control the encrypted data between the two endpoints of the communication channel, we need to ensure that users are also in control of 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 block specific traffic.

5 Evaluation

In this section, we present the setup and details about the implementation of the prototype, a use case, and results regarding performance and energy consumption.

5.1 Setup

We used two Raspberry PI 3 Model B+ (RP3): one representing a router and another to represent an IoT device. The IoT device has five sensor modules: Temperature and Humidity (DHT11), Luminosity (KY-018), LED (KY-016), a Heart-Rate sensor (SEN-11574) and a GPS (Adafruit Ultimate GPS Breakout).

These devices can simulate some personal assistants, such as Google Home or Amazon Alexa, as well as smart devices that connect to them, such as LEDs or thermostats. 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.

We use power to connect the RP3s instead of extra batteries (for example, power banks). To measure energy consumption, we used a direct plug-in energy meter [10].

5.2 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, as well as the integration between the sell data component and marketplaces.

Sharing Decision Control
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, even if the users specify a domain name as part of a rule, the DNS lookup will be done once, and the tool will create a new rule for the resulting IP address. Therefore, it is impossible to block any connection from a full domain name, including all subdomains. Besides, 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* [23], which provides a large number 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 [27], which has an option to decrypt requests transmitted using the HTTPS protocol called SSL Bump. However, as HTTPS provides security and privacy, decrypting these HTTPS tunnels without consent may violate ethical standards and be illegal, depending on the jurisdiction [28]. From a practical point of view, in this scenario, we have data similar to an ISP; if we use the proxy, no communication will be secret, making the system incompatible with the General Data Protection Regulation (GDPR) with all messages exchanged in *WhatsApp* or *Google*, for example, exposed on the local and remote machine.

For these reasons, we choose a firewall solution based on the SNI. By default, Transport Layer Security (TLS) [6] connections do not provide a mechanism for a client to tell a server the name of the server it is contacting. The header containing the Fully Qualified Domain Name (FQDN) [35] is encrypted, so we are unable to access that name. To overcome this problem, we choose 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 [34]. In addition, we use a combination of *SNIdump* (similar to TCPdump) and *whois*, to provide the user with real-time information.

Context-aware Internal Sharing

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 [33, 29, 21], especially related to self-learning techniques in IoT and decision making.

In this work, we combine some concepts already introduced in other related works to achieve the goal of defining different identities and access control policies for a device. The initial goal is to understand the context of the device 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*, to get their benefits (for example, opening the user's home window) [18]. 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.

Authentication

For security and testing purposes, the system uses certificates (public and private keys) where the router's owner is responsible for creating the Certificate Authority (CA) and distribute the certificates.

The system uses Elliptic Curve Digital Signature Algorithm (ECDSA) 256-bit to generate the CA and an Elliptic Curve Integrated Encryption Scheme (ECIES) 256-bit key pair to generate a shared key without the need for the Diffie-Hellman exchange. Elliptic Curve Cryptography (ECC) [15] can provide the security level relatively equal to the RSA with a smaller key [13].

We developed a Python implementation, requiring the *pycrypto* library. The *Crypto.PublicKey.ECC* module provides facilities for generating new ECC keys, allowing them to be rebuilt from known components, exported and imported. We use the Elliptic Curve Diffie Hellman Ephemeral (ECDHE) [19] algorithm

for client authentication and the implementation is based on two existing implementations of ECC and ECIES [3, 5].

Sell Data
This concept focuses on the control of 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.* [38]. 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 Results

Regarding performance, we measure the latency that a request/response takes to reach our device if it has to go through a certain number of *iptables* rules. For that, we use *iPerf* [8], an open-source program that generates traffic to allow performance tests on networks to measure bandwidth and link quality between two points. Besides, it measures delay jitter. We generate *iptables* rules with random IPs that do not match the tested client's IP (therefore, the client's request needs to test all the rules).

We tested with 0, 50, 500, 5000 and 50000 rules, and we present the results of UDP jitter (in milliseconds (ms) with standard deviation (sd)) and TCP bandwidth (in Mbits per second with sd) on the Table 1.

Num. Rules	**0**	**50**	**500**	**5000**	**50000**
TCP bandwidth (Mbits/sec ± sd)	26.7 ± 1.45	26.4 ± 1.84	25.9 ± 2.65	16.5 ± 0.78	3.35 ± 0.48
UDP jitter (ms ± sd)	0.43 ± 0.29	0.38 ± 0.42	0.46 ± 0.29	0.43 ± 0.41	4.98 ± 1.29

Table 1. *iPerf* measures

There is no significant difference in jitter and bandwidth using 0 or 5000 rules. These results show that the *iptables* tool is efficient, and that, despite the increased jitter and decreased bandwidth, when the number of rules is much higher (50000 rules), the solution remains viable and, therefore, the proposed solution can handle a large number of rules.
In terms of energy consumed, the RP3 spends $3.8W$ without services running on the system. With middleware and a database, the energy consumption remains the same. When the middleware performs encryption and decryption, the energy spent is not significantly different, varying only $0.6W$ up and down.

6 Security Analysis

This section overviews a threat model and some attack scenarios to perform a security analysis of our proposal.

6.1 Threat Model

The owner distributes the certificates to trusted devices; therefore, the owner is responsible for the safety of that 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, 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 Man-in-the-Middle (MitM) will not work as HTTPS certificates will not match. Besides, as already described in Section 5.2, the user has to manually intervene when there is a coordination failure in the ambient sensors or based on historical usage patterns.

6.2 Attack Scenarios

In this section, we present and evaluate some attack scenarios regarding the system implementation. We used some set of theorems adapting those used by *Afifi et al.* [1] and we added more that we find essential for this approach.

Claim 1. Security against tag impersonation attacks.
As already mentioned, the owner is responsible for distributing certificates to trusted devices, making that security anchored on the owner.
For filtering traffic and prevent MitM attacks, we chose the *iptables* firewall solution. First, contrarily to the proxy solution, we do not need to do MitM on HTTPS communication to decrypt it, which would make us susceptible to MitM, as SSL bump already does this type of attack on its own.
SNI also constitutes some impersonation problems [30]. 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.

Claim 2. Security against replay attacks.
HTTPS has a server-chosen random number, which is the server's first response in the handshake sequence [11]. If an attacker captures and tries to resend a message, the server must never allow it due to duplicate nonce. As nonce ensures that the server cannot reuse old communications, this method protects the replay attacks.

Claim 3. Resistance to Single Point of Failure.
On white-box devices, users can choose the data to be saved on the router or each of their 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, in the event of an attack, it only compromises one user. 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.

7 Conclusions and Future Work

We offer a novel middleware to improve the privacy of users' data on the IoT, with implementation and evaluation. Middleware gives users the ability to control the privacy of their data. Also, users can 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.
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 fully control the shared data.
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.
Based on our evaluation, we can state that *iptables* are viable in the implementation, presenting only differences in performance for 50000 rules.
As future work, we will create a real environment with a router with *OpenWRT* running the middleware that we created and with black-box devices together with the Raspberry PI's. 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 GDPR.

Acknowledgements. This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project UIDB/50014/2020. Patrícia R. Sousa's work was supported by a scholarship from the Fundação para a Ciência e Tecnologia (FCT), Portugal (scholarships number SFRH/BD/135696/2018). This work has also been supported by the EU H2020-SU-ICT-03-2018 Project No. 830929 CyberSec4Europe

(cybersec4europe.eu) and by National Funds through the Agência para a Modernização Administrativa, program POCI - Programa Operacional Competitividade e Internacionalização, within project POCI-05-5762-FSE-000229.1

References

1. Afifi, M.H., Zhou, L., Chakrabartty, S., Ren, J.: Dynamic authentication protocol using self-powered timers for passive internet of things. IEEE Internet of Things Journal **5**(4), 2927–2935 (2017)
2. Al-Hasnawi, A., Lilien, L.: 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. pp. 145–150 (2017)
3. Buchanan, B.: Elliptic curve integrated encryption scheme (ecies) with rabbit (2019), https://asecuritysite.com/encryption/ecc2
4. Conti, M., Dehghantanha, A., Franke, K., Watson, S.: Internet of things security and forensics: Challenges and opportunities (2018)
5. Corbellini, A.: Elliptic curve cryptography: a gentle introduction. https://github.com/andreacorbellini/ecc (2015)
6. Dierks, T., Rescorla, E.: The transport layer security (tls) protocol version 1.2. RFC 5246, August (2008)
7. Drozd, O., Kirrane, S.: I agree: Customize your personal data processing with the core user interface. In: International Conference on Trust and Privacy in Digital Business. pp. 17–32. Springer (2019)
8. Dugan, J.: Iperf tutorial. Columbus: Summer JointTechs pp. 1–4 (2010)
9. Eastlake, D.E.: Transport layer security (tls) extensions: Extension definitions. RFC **6066**, 1–25 (2011)
10. Efergy: Efergy home energy monitors: Electricity usage power monitor. https://efergy.com/about-efergy/ (2006)
11. Forouzan, B.A.: Cryptography & network security. McGraw-Hill, Inc. (2007)
12. Goddard, M.: The eu general data protection regulation (gdpr): European regulation that has a global impact. International Journal of Market Research **59**(6), 703–705 (2017)
13. Gueron, S., Krasnov, V.: Fast prime field elliptic-curve cryptography with 256-bit primes. Journal of Cryptographic Engineering **5**(2), 141–151 (2015)
14. Haar, C., Buchmann, E.: Fane: A firewall appliance for the smart home. In: 2019 Federated Conference on Computer Science and Information Systems (FedCSIS). pp. 449–458. IEEE (2019)
15. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of computation **48**(177), 203–209 (1987)
16. Lastdrager, E., Hesselman, C., Jansen, J., Davids, M.: Protecting home networks from insecure iot devices. In: NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium. pp. 1–6. IEEE (2020)
17. Lopez, J., Rios, R., Bao, F., Wang, G.: Evolving privacy: From sensors to the internet of things. Future Generation Computer Systems **75**, 46–57 (2017)
18. de Matos, E., Tiburski, R.T., Amaral, L.A., Hessel, F.: 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). pp. 1711–1715. IEEE (2018)

19. Nir, Y., Josefsson, S., Pegourie-Gonnard, M.: Elliptic curve cryptography (ecc) cipher suites for transport layer security (tls) versions 1.2 and earlier. Internet Requests for Comments, RFC Editor, RFC **8422** (2018)
20. OpenDNS, L.: Phishtank: An anti-phishing site. https://www.phishtank.com (2016)
21. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.: Context aware computing for the internet of things: A survey. IEEE communications surveys & tutorials **16**(1), 414–454 (2013)
22. Pihole: Pi-hole® network-wide ad blocking (2015), https://pi-hole.net/
23. Prince, M.: Technical details behind a 400gbps ntp amplification ddos attack. Cloudflare, Inc **13** (2014)
24. Ren, J., Dubois, D.J., Choffnes, D., Mandalari, A.M., Kolcun, R., Haddadi, H.: Information exposure from consumer iot devices: A multidimensional, network-informed measurement approach. In: Proceedings of the Internet Measurement Conference. pp. 267–279 (2019)
25. Research, N.P.R..E.: The smart audio report. https://n.pr/2zEk4UE (2017)
26. Roesch, M.: Snort - lightweight intrusion detection for networks. In: Proceedings of the 13th USENIX Conference on System Administration. p. 229–238. LISA '99, USENIX Association, USA (1999)
27. Rousskov, A.: Squid: Optimising web delivery (2013), http://www.squid-cache.org/
28. Rousskov, A., Tsantilas, C.: Squid-in-the-middle ssl bump (2019), https://wiki.squid-cache.org/Features/SslBump
29. Sezer, O.B., Dogdu, E., Ozbayoglu, A.M.: Context-aware computing, learning, and big data in internet of things: a survey. IEEE Internet of Things Journal **5**(1), 1–27 (2017)
30. Shbair, W.M., Cholez, T., Goichot, A., Chrisment, I.: Efficiently bypassing sni-based https filtering. In: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM). pp. 990–995. IEEE (2015)
31. Sivaraman, V., Gharakheili, H.H., Vishwanath, A., Boreli, R., Mehani, O.: 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). pp. 163–167. IEEE (2015)
32. Sørensen, D.A., Vanggaard, N., Pedersen, J.M.: Automatic profile-based firewall for iot devices. https://bit.ly/3fjjCdA (2017)
33. Sukode, S., Gite, S., Agrawal, H.: Context aware framework in iot: a survey. International Journal **4**(1) (2015)
34. Svee, N.A.: Filter tls traffic with iptables. https://github.com/Lochnair/xt_tls (2016)
35. Volz, B.: The dynamic host configuration protocol for ipv6 (dhcpv6) client fully qualified domain name (fqdn) option. RFC **4704**, 1–15 (2006)
36. Wang, H., Zhang, Z., Taleb, T.: Special issue on security and privacy of iot. World Wide Web **21**(1), 1–6 (2018)
37. Yang, C., Huang, Q., Li, Z., Liu, K., Hu, F.: Big data and cloud computing: innovation opportunities and challenges. International Journal of Digital Earth **10**(1), 13–53 (2017)
38. Zhao, X., Sajan, K.K., Ramachandran, G.S., Krishnamachari, B.: 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). pp. 270–271. IEEE (2020)