Privacy implications of switching ON a light bulb in the IoT world

Mathieu Thiery, Vincent Roca, Arnaud Legout

Mathieu THIERY*, Vincent ROCA, and Arnaud LEGOUT

Privacy implications of switching ON a light bulb in the IoT world

Abstract: The number of connected devices is increasing every day, creating smart homes and shaping the era of the Internet of Things (IoT), and most of the time, end-users are unaware of their impacts on privacy. In this work, we analyze the ecosystem around a Philips Hue smart white bulb in order to assess the privacy risks associated to the use of different devices (smart speaker or button) and smartphone applications to control it. We show that using different techniques to switch ON or OFF this bulb has significant consequences regarding the actors involved (who mechanically gather information on the user's home) and the volume of data sent to the Internet (we measured differences up to a factor 100, depending on the control technique we used). Even when the user is at home, these data flows often leave the user's country, creating a situation that is neither privacy friendly (and the user is most of the time ignorant of the situation), nor sovereign (the user depends on foreign actors), nor sustainable (the extra energetic consumption is far from negligible). We therefore advocate a complete change of approach, that favors local communications whenever sufficient.

Keywords: IoT, Connected Devices, data paths, smart home, privacy

1 Introduction

The number of connected devices is constantly increasing, with 20 billion devices forecasts for 2020 [1], and connected devices are making their way into an increasing number of houses. Once instrumented with connected devices, these houses can become a breeding ground for data. In these so-called smart homes, a wide range of devices can be found, each of them capturing data (e.g., temperature, voice, or live video). Additionally, chances are that in a smart home, most of the captured data will be personal by essence – i.e., linked directly or indirectly to a physical person. It means that the European General Data Protection Regulation (GDPR) [2] will apply to any company that collects data from European Union residents.

However, companies' honesty aside, producing privacy preserving services is not an easy task. When designing a connected device, the manufacturer needs to deal with multi-layered problems concerning hardware, software, data storage, transmission, encryption, anonymization, and user's informed consent collection, to cite a few ones. Therefore one can expect many issues to arise from bad product designs, be it from a private company or from open-source projects. Furthermore, the IoT domain requires that several *de facto* standards or connected devices from different manufacturers coexist and interoperate seamlessly, which is challenging in such a highly dynamic domain.

Hence, the more devices are introduced in a smart home, the more smartphone applications are used to control them, the more confusing it gets, especially when they come from different manufacturers, a usual situation.

This work aims at assessing the control part of the target connected device we chose, namely a Philips Hue white bulb [3], using several techniques to switch it ON or OFF. When at home, we assume the user uses either a smartphone that is connected to the home network (i.e., the smartphone is connected through the local Wi-Fi access point), a smart speaker, or a physical smart button. Several applications are available on the user's smartphone that can be used to that purpose, coming from several developers or companies. When remote, the user uses his smartphone and either a 4G cellular connection or a connection to a visited Wi-Fi network, and one of the applications that can work remotely. The evaluation performed is essentially centered on privacy considerations, focusing in particular on the data paths: who is informed of what in the user's house?

Our main contributions are the following:

– This work adopts an original viewpoint whereby we focus on the control part of a given connected device, rather than the connected device itself: it

***Corresponding Author: Mathieu THIERY:** Univ. Grenoble Alpes, Inria, mathieu.thiery@inria.fr
Vincent ROCA: Univ. Grenoble Alpes, Inria, vincent.roca@inria.fr
Arnaud LEGOUT: Univ. Côte d'Azur, Inria, arnaud.legout@inria.fr

shows that major differences are caused by the technique being used to control it, and therefore many observations and conclusions remain valid across a large set of devices.

- It highlights major privacy and sovereignty concerns: for instance, in terms of outgoing data volume, it shows that the large majority (almost three quarters in total, summing up the traffic across all scenarios) of the user's personal data that leaves the house (situated in France) is transmitted, stored and processed on servers located in the US, the traffic remaining in European servers representing just above a quarter. In terms of data paths, the analysis of the ON/OFF requests also highlights seven different categories, with largely varying implications for the end-user's privacy.
- Then it shows that inferring the user's action by analyzing the request size is rather efficient, even with encrypted traffic.
- In any case, it is obvious that the situation is too complex for the end-user to be in position to clearly understand the privacy implications of the various choices available to him to control his smart bulb. This is a major concern when the GDPR requires a clear and informed consent before a data controller can collect and process any personal data.

The paper is organized as follows. We first describe the experimental methodology and the tools designed to that purpose. We continue with an analysis of experimental results, focusing on the data flows, the possible inferences in particular through the message size analysis, on sovereignty aspects, and the particular case of a remote user. We finish with a discussion, a review of related work, and a conclusion.

2 Methodology

2.1 Goals of the Experiments

Our experimental environment aims at evaluating the control part of the target connected device we chose, namely a Philips Hue white bulb. The choice of this target device is first of all related to its popularity, making it a device of choice in many smart homes. This popularity also means that this device is interoperable with a large variety of control techniques, a key requirement in our case. The nature of the device, a smart bulb, is also interesting, as many end-users under-estimate its privacy implications (e.g., compared to a camera with facial recognition capability), whereas it could easily be used to infer the house inhabitants' habits, telling the difference between the regular versus exceptional patterns. Finally, the device itself does not produce a significant amount of data, meaning that the data volumes we observe are mainly caused by the control part itself.

The experiments focus on two different situations. First of all, we consider a user at home, using either a smartphone (or tablet) that is connected to the home network (i.e., the smartphone is connected through the local Wi-Fi access point rather than through a 4G cellular connection), or using a smart speaker or smart button. Secondly, we consider a remote user, using his smartphone and 4G cellular connection. Here the goal is to assess whether this remote control is feasible or not, and to appreciate the consequences in terms of communications. One of the questions we are interested in is whether a company over-privileged the ability to remotely control a bulb, a non essential feature, forgetting to have a purely local control when this is sufficient, as data minimization is a key privacy feature, or in case of Internet connection outage, a key practical requirement.

2.2 Testing Environment

Table 1 lists the various physical components of the platform, including their firmware or operating system version (since a device behavior may change across versions).

Table 1. List of used physical devices, with their firmware version and connectivity technology. *The Raspberry Pi 3 Model B is used for both Home Assistant and OpenHABian software.*

Device	Firmware version	Connect.
Amazon Echo Spot [4]	625533420	Wi-Fi
Google Home [5]	137090	Wi-Fi
IKEA Tradfri Gateway [6]	1.8.26	Eth. + ZigBee
IKEA Tradfri Remote Control [7]	1.2.223	ZigBee
LG Nexus 5 Smartphone [8]	Android 6.0.1	Wi-Fi
Philips Bridge [9]	1809121051	Eth. + ZigBee
	1932073040 [1]	Eth. + ZigBee
Philips Hue white Bulb [3]	1.29.0_r21169	ZigBee
Raspberry Pi 3 Model B [10]	Home Assistant 1.12 / OpenHABian 1.4.1	Wi-Fi

The smartphone contains several applications to control the smart bulb (see the list in Table 2): Amazon Alexa is the official application to manage the Echo Spot and has its own interface to switch ON and OFF the Philips bulb; Same thing goes for the Google Home application; All4Hue and Hue Hello are two non official applications that are meant for the Philips ecosystem specifically; IFTTT is an application meant for IoT in general that embeds a variety of widgets, including one that switches ON and OFF a Philips bulb; OpenHAB is an open-source application that connects directly to an open-source OpenHAB server hosted on a Raspberry Pi; Home Assistant is another open-source solution that is hosted on a Raspberry Pi. It does not have any official application on Android, but the Home Assistant server hosted on the Raspberry Pi is accessible from any browser (we used Chrome browser in our experiment).

Table 2. List of used smartphone applications. *We didn't find any official application for Home Assistant on Android which is why we used Chrome to access the Home Assistant web interface on the Raspberry Pi.*

Application	Version
Amazon Alexa [11]	2.2.241878.0
All4Hue [12]	8.8
Google Home [13]	2.7.21.2
Chrome (for Home Assistant) [14]	70.0.3538.80y
Philips Hue [15]	3.9.0
Hue Hello [16]	0.99.99.72
IFTTT [17]	3.7.4
OpenHAB [18]	2.2.0
IKEA Tradfri [19]	1.9.1

In addition to these devices, the experimental setup depicted in figure 1 includes a laptop (Dell Latitude E6410 running Linux) acting as the box of an Internet service provider: it relays traffic to/from Internet, it provides the `wlan0` Wi-Fi interface (it uses a Penguin Wireless USB TPE-N150USB Wi-Fi adapter in Access Point mode), and the `eth0` wired Ethernet interface. The Wi-Fi `wlan0` and Ethernet `eth0` interfaces are both used to provide connection on the home LAN. The `eth0` interface is connected to an Ethernet hub (rather than a switch) in order to enable Ethernet interface traffic sniffing on the various ports of the simulated home box, and a Wi-Fi access point with its own SSID is defined on the `wlan0` interface.

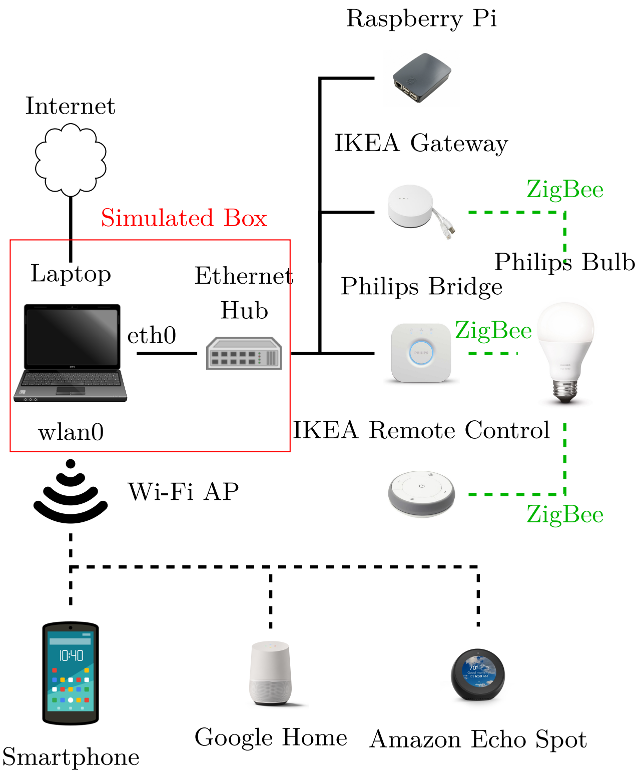


Fig. 1. Core setup. *The laptop is used to relay the communications between the Internet and the devices connected to the Ethernet hub or the Wi-Fi access point (managed by the laptop itself). The combination of the laptop, hub and access point simulates an ISP box.*

To deal with the Internet relaying and have the two networks (wired and wireless) from `eth0` and `wlan0` on the same subnet, we used a mix of `iptable` and software bridge rules, as well as specific `wpa_supplicant`, `hostapd` and `dhcpd` configurations. This setup (see section 2.8) enables to connect devices on the hub or on the access point, and have them communicate seamlessly with each other and with the Internet.

This testbed enables to capture all the traffic (local, to and from the Internet) using Wireshark on the laptop. The only exception is the ZigBee network between the Philips bridge or the IKEA gateway and the Philips bulb which is not monitored. This is not an issue as we are mainly interested in the Internet traffic rather than the local traffic between these two devices.

[1] Our first bridge was broken during the experiment. We bought a new one and couldn't downgrade its firmware to the old version. After some tests to confirm that the behaviour of the new one was identical to the old one, we used this new bridge for the experiment related to the IKEA devices and application only.

2.3 Baseline versus In-Use Captures and Methodology

All the captures fall into two categories: "baseline" or "in-use". These categories are used to distinguish between the default network traffic (baseline captures) and the traffic generated by switching ON or OFF the Philips bulb (in-use captures). All the captures were made from the `Any` interface [2] available on Wireshark in order to see every communication going on, be it from device to device or from device to the Internet and vice-versa.

We show in figure 2 the methodology used for these baseline and in-use captures. The third step ("wait for 10 minutes") is required to ensure that all devices reach a stable state after being switched ON (it can take around 10 minutes for some of them).

For baseline captures, the 60 seconds waiting time (step 5) was chosen so as to keep a reasonable total capture time and still seeing the not so frequent but recurrent frames that can be sent by a device. For in-use captures, we switch ON and OFF the bulb two times (i.e., ON – OFF – ON – OFF) to increase the accuracy of measurements and reduce the risks of missing the not so frequent recurrent frames.

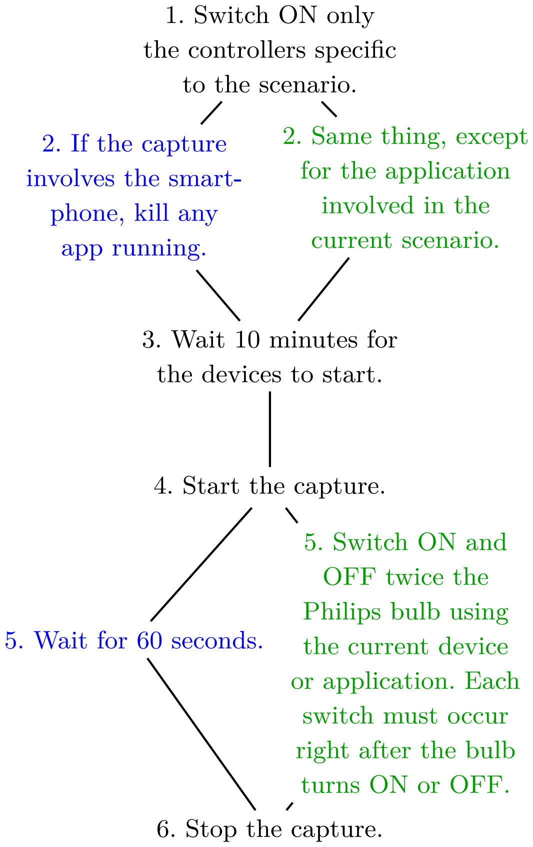

1. Switch ON only the controllers specific to the scenario.
2. If the capture involves the smartphone, kill any app running.
2. Same thing, except for the application involved in the current scenario.
3. Wait 10 minutes for the devices to start.
4. Start the capture.
5. Wait for 60 seconds.
5. Switch ON and OFF twice the Philips bulb using the current device or application. Each switch must occur right after the bulb turns ON or OFF.
6. Stop the capture.

Fig. 2. Captures Methodology. *On the left (blue) is the methodology used for the baseline captures, on the right (green) the one for the in-use captures, in the middle (black) are the common steps.*

2.4 The 21 Baseline and 14 In-Use Scenarios

The methodology described in figure 2 has been applied to various scenarios, each of them related to a specific manner to control the Philips bulb (in-use capture), or serving as a reference (baseline capture). Hence we performed one capture per scenario[3].

Table 3 lists the baseline scenarios, and for each device three scenarios (i.e., three captures): device alone, device plus the Philips bridge, and device plus the Philips bridge and bulb. With IKEA devices (Tradfri gateway and remote control), we assume the user adds them to its existing configuration without changing this later as he may use other techniques that require the Philips bridge. The case of the IKEA remote control is specific in the sense this smart button communicates directly to the bulb in ZigBee, making baseline scenarios with the button alone or associated to the Philips bridge alone meaningless. In total, 21 scenarios are considered.

Concerning in-use scenarios, table 4 shows the list of devices and smartphone applications we considered as well as how they are used: directly (i.e., smart speakers and IKEA remote control), by using the smartphone microphone through the application, or by using an ON/OFF button in the application. In total, 14 scenarios are considered.

The baseline captures are used as references to distinguish the background traffic (with or without the two Philips devices) from the in-use captures that also include the specific traffic generated during bulb control.

It is important to note that Home-Assistant and OpenHAB solutions both require a Raspberry Pi (web interface accessible by the smartphone through a browser with Home-Assistant, or via a dedicated smartphone application with OpenHAB). So the captures made for these scenarios are actually collecting the data

[2] Wireshark allows captures on every interface and provides an interface called `Any (Linux cooked)` that merges them into a single one.

[3] We repeated these captures in some circumstances, e.g., in Section 3.3 to increase confidence in the ON and OFF frame sizes.

Table 3. The 21 baselines scenarios. *For every device, we defined three scenarios: device alone, device plus Philips bridge, and device plus Philips bridge and bulb. The three situations for which a baseline scenario makes no sense are marked "n/a".*

Device	alone	Philips bridge	bridge and bulb
None	n/a	✓	✓
Amazon Echo Spot	✓	✓	✓
Google Home	✓	✓	✓
Smartphone	✓	✓	✓
Raspberry Pi (Home Assistant)	✓	✓	✓
Raspberry Pi (OpenHAB)	✓	✓	✓
IKEA Tradfri Gateway	✓	✓	✓
IKEA Tradfri Remote Control	n/a	n/a	✓

Table 4. The 14 in use scenarios. *Here, the smartphone is repeated several times as it is used for several applications. In cases of Home Assistant and OpenHAB, the scenario involves both the smartphone and a Raspberry Pi that runs the associated software.*

	Control technique		
Devices	Amazon Echo Spot	✓	
	Google Home	✓	
	IKEA Tradfri Remote Control	✓	
		micro	button
Applications	Smartphone + Alexa app	✓	✓
	Smartphone + All4Hue app	n/a	✓
	Smartphone + Home app	✓	✓
	Smartphone + R. Pi + Home Ass.	n/a	✓
	Smartphone + Philips Hue app	n/a	✓
	Smartphone + Hue Hello app	n/a	✓
	Smartphone + IFTTT app	n/a	✓
	Smartphone + R. Pi + OpenHAB	n/a	✓
	Smartphone + IKEA Tradfri app + gw	n/a	✓

from the bridge, the smartphone and also the Raspberry Pi. The Raspberry Pi is otherwise not used in the other scenarios.

2.5 Our Statistics and Labeling Framework

In order to easily understand what is actually going on in the network, we developed tools to compute statistics on the captures. Our `pcapstat` Python tool (this tool is publicly available, see section 2.8) performs a capture and summarizes the results across several parameters, such as for instance, the frames count, the manufacturer of the devices, or the source and destination. It tries to automatically deduce some information like the identity of a source or destination using the `whois`, `geoiplookup` and `host` tools, and by exploiting DNS requests that are detected. Every source and destination is labeled in a human-readable way and all the data generated by `pcapstat` is exported and ready for post-processing using the Pandas framework [20].

The label we use for sources and destinations is what we call the "best known label". It consists of an IP address in the worst case situation, otherwise, the more data we managed to deduce, the more we improve the label. Several pieces of information are potentially used for this label, namely the `whois` of the IP address, the country code of the server, the server host name (its "FQDN") returned by the `host` command, or the exact name if any DNS request was found in the capture to that particular IP.

Here are some examples of labels generated:

1. `(US) AT-88-Z/ ip: 52.95.121.5 (IP/TCP)`
2. `(US) GOOGLE/ rdns: time1.google.com (IP/UDP/NTP)`
3. `(US) GOOGLE/ drdns: www.google.com (IP/TCP)`

The first example is for the case with only an IP, a country, an organization ("Amazon Technologies"), and a protocol: we only managed to get a result from the `whois` and `geoiplookup` commands, nothing else. The second example adds an `rdns` keyword (Reverse DNS), which means that we managed to get a result from the `host` command, thus replacing the IP by a host name. The third example gives what we called `drdns` (Detected Reverse DNS) which means that we found a DNS request inside the capture itself to that host name, which improves reliability (indeed, several host names for different services can be associated to the same IP address, creating an ambiguity unless the reverse DNS request is intercepted).

Having meaningful labels and automated extractions across several viewpoints turned out to be key in the analysis of the numerous and complex captures.

2.6 Accessing Packets' Cleartext

We were also interested in the packets content. When the packets are in cleartext, we dump their content, removing any non-readable ASCII character. Fortunately from a security and privacy viewpoint, many packets are encrypted. In that case, we launch a man-in-the-middle attack, using several tools.

In case of smartphone applications, the first tool is the `Burp Suite` [21], that intercepts frames sent by an application and decrypts them when possible. The smartphone user needs to set Burp as a proxy in the smartphone's "Internet settings" and add Burp's certificate authority to the Android trusted certificates. After that, Burp is trusted by Android, it can generate its own certificate in place of a frame destination server and take the role of the application towards the server. This way, each frame goes through Burp and gets decrypted, in a transparent manner. Burp also allows to drop or forward any packet it receives, which is convenient to identify which packet triggers the ON or OFF action on the bulb.

However, certain applications use certificate pinning and will not trust any certificate that is not defined in their code. To circumvent this, we used the Frida tool, along with a script [22] meant to disable certificate pinning. Thanks to that script, we were able to access the packet's cleartext of Alexa, the only application among those we considered that used certificate pinning.

Using these techniques, we were able to get the packets content of each application we considered, except for the official Philips Hue application and the IKEA Tradfri application. The reasons for those failures are unclear, but it could be due to proxy detection or bypassing mechanisms, or to the use of DTLS instead of HTTPS for Tradfri.

Concerning the connected devices (in particular the Philips bridge), we failed at accessing their packets content. Indeed, Burp requires the user to add a new certificate authority, that of Burp, in the device. This is possible on the Android smartphone, using the user configuration facilities, but not in the connected devices we were using. Hence no packets could be decrypted for those devices.

2.7 Ethical Concerns

All the captures conducted in this experiment were done in a controlled environment, in a private encrypted WPA2 network, using made-up usernames and e-mails, without any actual user being involved. Thus we can assure that no personal data was gathered that could be linked to any physical person.

2.8 Research Reproducibility

All the tools used and all the raw captures performed in this work will be publicly released prior to publication. All the configuration details discussed in Section 2 will be publicly disclosed prior to publication.

3 Experimental Results

This section details the experimental results achieved. We start with a macroscopic analysis, considering the data paths, the volume of data exchanged on the Internet, and sovereignty considerations. In a second step we look at the messages themselves, how their size often enables to infer the user actions on the bulb, their content, as well as the presence of trackers in most of the smartphone applications used. We finish with an analysis of features beneficial to privacy and assess whether or not they are used in the various techniques considered for the control part.

3.1 Taxonomy of Data Paths

3.1.1 About Baseline Scenarios

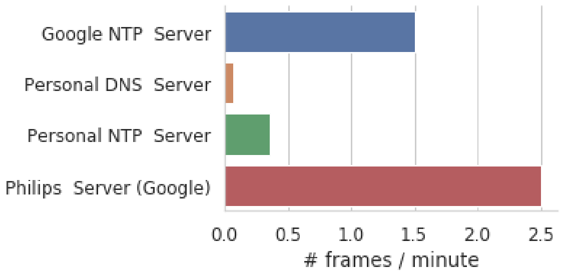


Fig. 3. Average number of frames per destination sent by the Philips bridge across all baseline scenarios involving the bridge. *Google is clearly the main actor because of Philips choice to rely on Google services to host the Philips server.*

Let us first focus on baseline scenarios. In all scenarios involving the Philips bridge, recurrent communications happen between the Philips bridge and the Philips server, hosted by Google. Figure 3 shows that frames are sent approximately 2.5 times per minute, on average. These communications may be used to update the bulb status although we have no way to prove it, the traffic being encrypted with no solution to access the cleartext. However, an other reasonable explanation is that those communications are just used as a keep-alive mechanism.

We also assess the reference behaviour of each physical device taken with no Philips bridge. We observed in particular that the IKEA Tradfri gateway does not generate any traffic to the Internet (NTP or DNS included) as opposed to the Philips bridge.

3.1.2 About In-Use Scenarios

Let us now focus on in-use scenarios. In order to understand how communications happen, we analyzed their data paths during the ON or OFF requests. Figure 4 lists the five categories of paths observed:

- Category 1 is for the Google Home smart speaker or smartphone application. Here the raw audio capture is first sent to the Google Web API (remote) server for Natural Language Processing (NLP). Then the associated request is generated and reaches the Philips (remote) server, also hosted by Google, but this part of the communication taking place in the Internet, the details are not visible. From this point, it reaches the Philips bridge where it triggers the action on the bulb. At the same time, the bridge also communicates back to the Philips Server. Finally, after some delay, a periodic communication happens from the Philips bridge to the Philips server that could be either a bulb state update and/or a keep-alive mechanism (see below).
- Category 2 is for the Amazon smart speaker or smartphone application. It is somewhat similar to category 1, with the exception that the Web API server that performs NLP is hosted by Amazon rather than Google. The associated request is generated and reaches the Philips server. The remaining of the exchange is identical to category 1.
- Category 3 is for the IFTTT smartphone application. The observed behavior is close to that of category 2, i.e., the request travels through Amazon before reaching Google.
- Category 4, used by the three applications mentioned, involves purely local communications by default (although the Philips Hue application can also go through its Web API server if needed, e.g., when connected from outside of the house). The remaining of the exchange is identical to category 1.
- Category 5 is for the two open-source systems, Home Assistant and OpenHab, that both require the bulb action to reach the Raspberry Pi server, which then communicates directly to the Philips bridge. The remaining of the exchange is identical to category 1.
- Category 6 is for IKEA Tradfri application and gateway. The ON/OFF request leaves the smartphone and goes directly to the IKEA gateway. Then this gateway sends a ZigBee request to the bulb, thus bypassing the Philips bridge altogether. Later on, we still see recurrent communications from the bridge to the Philips server.
- Category 7 is the shortest: The IKEA Remote Control is triggered and immediately sends a ZigBee request to the bulb. Here also the Philips bridge is totally bypassed, but we still see recurrent communications from the bridge to the Philips server.

Several key aspects are highlighted by this first classification. We see that categories 1, 2, and 3 heavily rely on the availability of an Internet connection in order to use API Servers hosted by Google or Amazon. In all cases, requests are periodically sent by the Philips bridge to the Philips server hosted by Google[4], unlike the other categories. They may embed the state of the bulb and/or could be used as a keep-alive mechanism (e.g., in order to avoid the state created by NAT traversal mechanisms to time-out). Since this traffic cannot be decrypted (see Section 2.6), we cannot conclude.

Therefore we see that Google – and to a lesser extent Amazon – that host all the Web API Servers encountered, are central to this ecosystem and gather by design a lot of personal information.

Other categories demonstrate better behaviours by allowing local communications. OpenHAB and Home Assistant have the same approach by just relaying the request from the smartphone to the Philips bridge using a Raspberry Pi. The IKEA gateway (category 6) goes even further by completely bypassing the Philips bridge. This means that we can even just get rid of the bridge, and no data will be sent outside of the network according to our baselines of the gateway. Finally the IKEA Tradfri remote control (category 7) depicts the best behaviour possible: it bypasses the Philips bridge and doesn't even need the IKEA gateway. So using the remote control allows a user to switch on his bulb using ZigBee only without involving any other party.

In conclusion, the use of the IKEA Tradfri remote control seems to be the most appropriate way of using the Philips Bulb in terms of privacy but it will fail at accessing the bulb remotely. For remote access, the best would be to use a VPN along with the IKEA gateway.

[4] Categories 1 and 2 both contained one capture for which Amazon was also exceptionally hosting the Philips server.

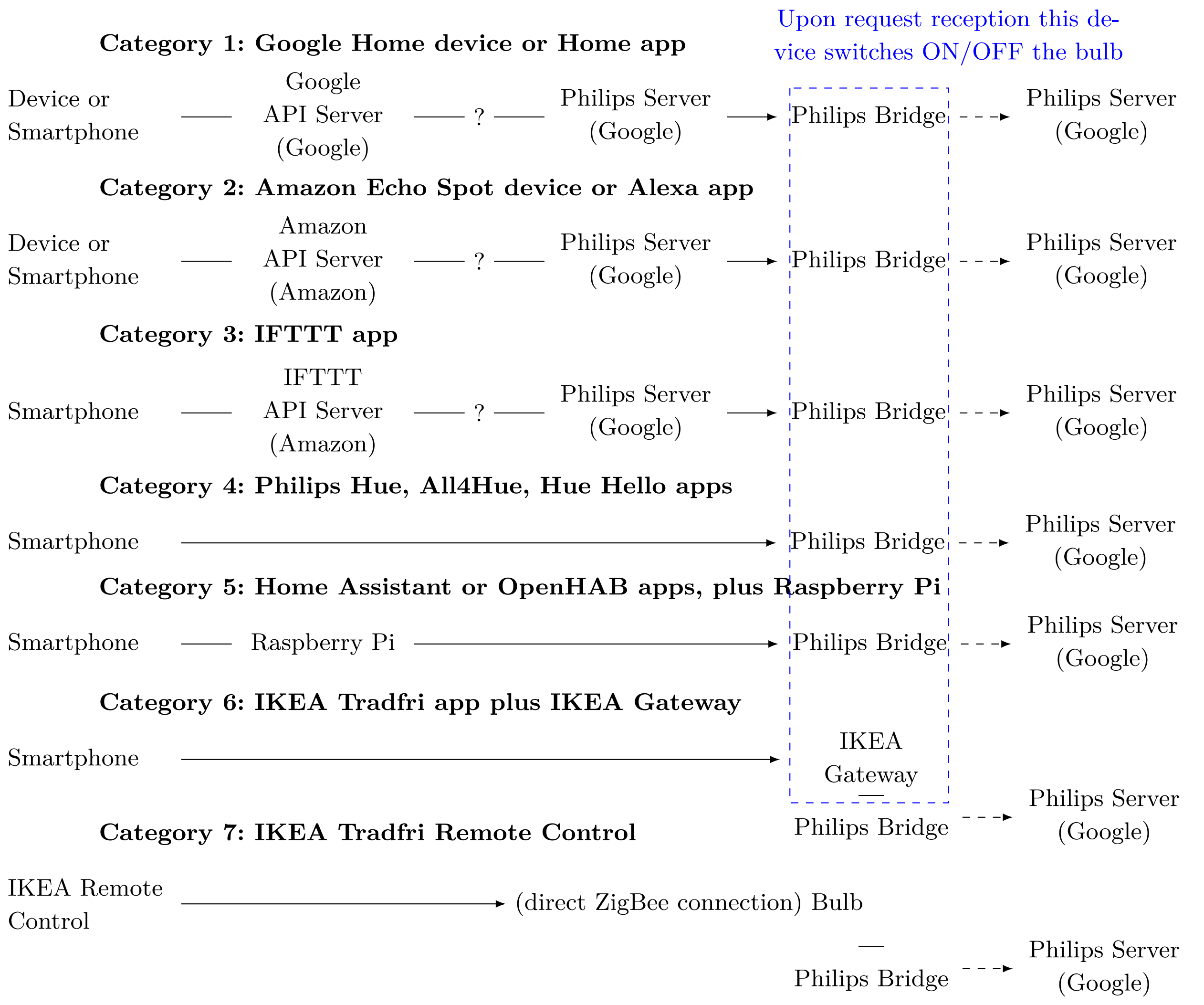


Fig. 4. Taxonomy of the ON/OFF request paths for the 14 in-use scenarios. *We identify 7 different categories, representative of the manufacturer strategies. In parentheses are the hosters of the server mentioned above. Question marks are for the hidden communications that happen in the Internet and are not visible to our monitoring tools. The dashed arrows refer to the traffic to the Philips Server triggered by an ON/OFF request (categories 1 to 5 included) and the recurrent communications that happen periodically (all categories) as highlighted in the baseline captures.*

Without a VPN, Home-Assistant or OpenHAB would be the best solutions, provided that they are setup to use HTTPS.

3.1.3 The Case of a Remote User

So far we considered the case of a user at home. We also performed experiments on how each control technique would behave if used from outside home.

The two smart speakers and the IKEA Remote Control are, by design, meant to stay home.

The All4Hue and Hue Hello applications both connect directly to the Philips bridge, using the local home network. They do not work if the smartphone is connected through 4G rather than Wi-Fi on the home network, and there is no suggested technique for those applications to connect from outside home.

Otherwise, there are three ways used to communicate between a remote smartphone and the Philips bulb:

- The local Philips bridge opens a connection with the remote Philips Server. When the user triggers an action, the request will eventually get to the Philips Server that will relay it to the bridge using the already existing connection. That's the technique used by the Home, Alexa, IFTTT and Philips Hue (when the user is outside home) applications.

- The IKEA Tradfri application by default requires a direct local connection to the IKEA gateway. However it can be used remotely as follows: the application and gateway can be associated to an Amazon Alexa or Google Assistant account thanks to a dedicated IKEA Tradfri "skill". This process creates a connection between the IKEA gateway and Amazon or Google servers, and this connections can be used by a remote smartphone to redirect its requests to the IKEA gateway and bulb.
- The user's ISP box is setup with a known IP address and a dedicated port is defined in it to forward traffic to this port number to the local device (i.e., the Raspberry PI). This way, a remote smartphone can connect to the Raspberry Pi through this open port number and control the smart bulb. This is the technique used by default by Home Assistant and OpenHAB, although both can use the first technique as well (e.g., the OpenHAB documentation recommends the use of their OpenHAB cloud).

Forcing the user to be dependent on external servers (Philips server, or Amazon or Google) is a privacy issue. This is why the use of port forwarding on the user's ISP box, although being a bit complex, is a better solution from a privacy point of view. However, it is important to notice that it is a dangerous solution from a security point of view, as an open port can also be an entry point for an external attacker.

3.2 About Sovereignty

Figures 5 (a) and (b) show the data volume sent to the Internet in each scenario, respectively by country and by hoster. We clearly see the hegemony of Google and Amazon, which explains also why US is the first country in terms of data collection in our captures. This traffic is partly due to their own services, but essentially to their hosting system heavily used by Philips. More precisely, when summing the outgoing traffic that leaves the house (situated in France), over all scenarios, we measured that 73.35% – i.e., almost three quarters – of it is destined to servers located in the US, the traffic remaining in European servers representing only 26.65%.

The four scenarios with the biggest amount of data sent to Google and Amazon are mainly due to microphone usage and Natural Language Processing (NLP) that is performed in the company's servers. This is visible both when looking at Google Home and Amazon Echo Spot network traffic, but also for the Home and Alexa smartphone applications when used with the microphone. We observe one to two orders of magnitude more traffic with these four scenarios than it is for the other applications, which also raises sustainability concerns (see Section 4.2).

The recurrent frames between the Philips bridge and the Philips server, even if they are just meant to keep the connection alive, are also concerning, as they give a permanent access to the bridge from outside the house, and in our case, from an other non-European country.

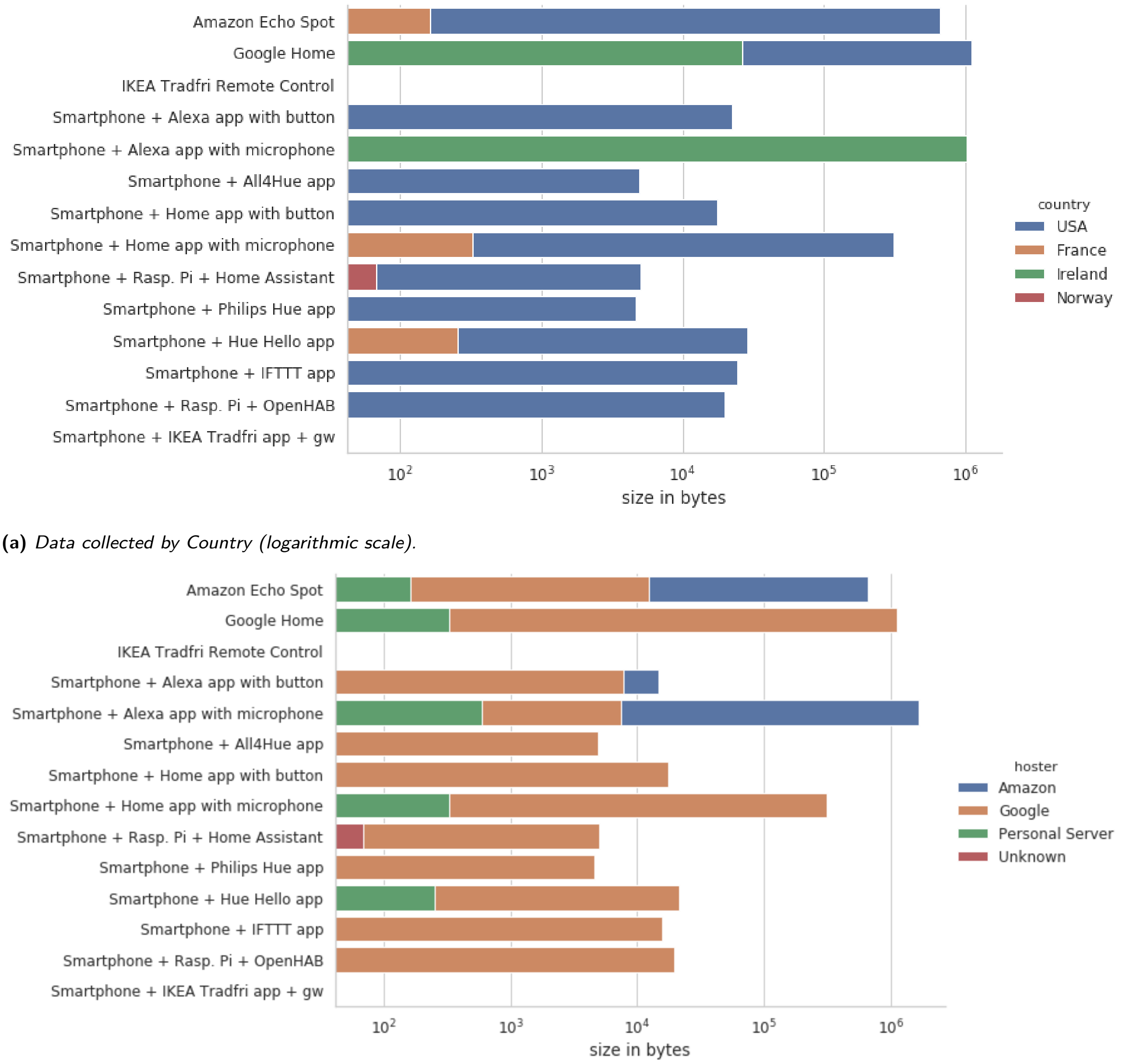
3.3 User Action Inference: Analysis of the ON/OFF Request Sizes

We have so far considered the data paths in general. We now focus on the messages themselves and start by showing that an analysis of their size often enables to infer the user action on the bulb, despite encryption.

Table 5. Action inference by frame size. *We can see that half of the captures revealed a 1 byte difference between the ON and the OFF requests. This is due to requests encoding: an "ON" versus "OFF" or "TRUE" versus "FALSE" strings. Any capture from which we couldn't extract a consistent frame size after 4 attempts is set to "fail" (failure). Captures involving devices or applications that are sending audio streams that prevented us from getting consistent frame sizes are set to n/a, as well as the IKEA remote control that directly communicates to the bulb.*

	Control technique	on size	off size	diff.
Encrypted	Amazon Echo Spot		n/a	
	Google Home		n/a	
	IKEA Tradfri Remote Control		n/a	
	Smartphone + Alexa app	269	270	1
	Smartphone + Alexa app with mic		n/a	
	Smartphone + Home app	245	246	1
	Smartphone + Home app with mic		n/a	
	Smartphone + Philips Hue app	258	259	1
	Smartphone + IFTTT app	fail	fail	fail
	Smartphone + IKEA Tradfri app + gw	fail	fail	fail
Cleartext	Smartphone + All4hue app	356	357	1
	Smartphone + R. Pi + Home Ass.	103	86	17
	Smartphone + Hue Hello app	229	230	1
	Smartphone + R. Pi + OpenHAB	275	276	1

Table 5 shows that it is often possible to determine if a message sent by a device or an application is an ON or OFF request, based on its size, even when the request itself is encrypted. This table has been elaborated



(a) *Data collected by Country (logarithmic scale).*

(b) *Data collected by Hoster (logarithmic scale).*

Fig. 5. Outgoing data volume for each in-use scenario. *This table classifies the outgoing traffic to Internet destinations according to the destination country or hoster. Note that the two Ikea scenarios do not generate by themselves any outgoing traffic to the Internet.*

thanks to the joint use of the Burp tool (with encrypted traffic) and traffic monitoring.

Any scenario that uses the microphone at some point is set to "n/a" (including when it is a smart speaker) because finding a frame size difference that is consistent over several trials was not possible (because of the microphone traffic).

From the 14 scenarios listed in the table, 7 ON and OFF requests could be inferred from the frame sizes. We can see that 6 captures are using OFF requests that differ from ON requests by only one byte. This one byte difference is easily explained by the use of the "ON" versus "OFF", or "TRUE" versus "FALSE" strings. We discovered this fact by looking at the content of the cleartext frames (using Burp when needed). The Home Assistant scenario has a 17 bytes difference because Home Assistant provides additional information in an ON request.

The case of IFTTT is particular, the ON and OFF requests being the same size (indicated as failure in the table). After inspecting the cleartext content of IFTTT requests using Burp, we saw that this is due to the fact that the content of ON and OFF requests are completely identical: they are just meant to ask the IFTTT API Server to switch the bulb's state.

The case of the IKEA Tradfri application is similar, with ON and OFF requests of the same size (indicated as failure in the table). However, because of the use of DTLS, we were not able to decrypt the requests and understand the underlying reason.

All these inferences have privacy implications: knowing when a bulb is ON or OFF enables to profile a user's activity very precisely. Of course, an attacker monitoring the message sizes only will not necessarily be in position to distinguish between the two types of requests (other messages may turn out to be of the same size), especially if the attacker does not know the exact nature of the bulb control technique. However this is a piece of information that can help inferring the user action, and as such, it contributes to the threat. In fact solving the problem is trivial since it is sufficient to have equal size requests, using either padding or a different encoding (e.g., "1" or "0" instead of "ON" or "OFF").

3.4 Messages Content and Presence of Application Trackers

3.4.1 About ON/OFF Requests

Interestingly, when looking into the actual content of the ON/OFF requests, these requests are mostly composed of fields for the ON or OFF status and for reporting errors. Except from the bulb status (which remains a personal information), nothing else has been found that would be concerning.

3.4.2 Privacy Leaks in Messages other than ON/OFF Requests

The situation is different for other messages, that are not ON/OFF requests. Table 6 gives a selection of fields present in these messages that may cause problems in terms of privacy and security:

- any type of identifier can allow user tracking;
- any type of information about the state of the device is a privacy leak as it can be used to get insights on the user activity;
- geolocation information is of course personal data;
- firmware or software versions can be used to easily find exploits and can thus lead to security problems;
- the ZigBee channel field tells an attacker what to monitor in the network;
- IP address, device name and model can be useful for network discovery;
- logins can lead to both privacy and security issues, firstly because it can embed personal information and secondly because it can lead to an unauthorized authentication;
- the whitelist field contains the list of all the applications that have been used by the user to interact with the Philips bridge, and this list, combined to other information like logins and version numbers, can also lead to privacy and security issues.

The first row concerns the local communication between the Philips bridge API and other applications. There are two issues with this local traffic. First, it is sent in clear-text, and second, these fields may be part of the Philips bridge recurrent frames that it sends periodically.

The second row concerns Alexa and clearly consists of personal information like geolocation sent directly to Amazon.

The third row concerns IFTTT and is composed of data collected by two trackers.

This brings us to the issue of trackers that are one of the most important vectors of personal information leakage.

3.4.3 Privacy Leaks in Application Trackers

Smartphone application trackers can also play a big role in personal data collection. Table 7 shows the list of trackers used by the selected applications, identified using Exodus Privacy [23]. We see here again that Google plays a key role. The OpenHAB application includes two trackers, however being open source, deriving a tracker free version of the application is feasible. The All4Hue and IKEA Tradfri applications are the only ones for which Exodus Privacy does not identify any (known) tracker. Finally, the Home Assistant case is special as there is no associated application at the time of writing (a web browser is used to connect to the Home Assistant web server on the Raspberry Pi).

A tracker free application is always beneficial from the privacy point of view, but there are other paramount features that a privacy-preserving application should implement. We discuss them next.

Table 6. Selection of fields for messages other than ON/OFF requests. *Hereby we report fields that may be used as identifiers, that are disclosing personal information, or could lead to security issues. They are classified according to the local versus remote communication nature. Other fields were found from Google Home application, that do not match our selection criteria. With Philips Hue and IKEA Tradfri, we were not able to access any cleartext content and therefore it is absent from this table.*

	From	To	Fields
Local	Philips bridge	All4Hue, Hue Hello, OpenHAB,	bridgeid, modelid, swversion, groups.*.name, lights.*.state.bri, lights.*.state.on, lights.*.swversion, lights.*.uniqueid
		All4Hue, Hue Hello, OpenHAB	apiversion, gateway, ipaddress, name, whitelist.*.name, zigbeechannel, scenes.*.name
Remote	Alexa	kinesis.us-east-1.amazonaws.com	app_id, version_name, client_id, country, device.make, device.model, platform.version, event_timestamp
		latinum-eu.amazon.com	latitudeInDegrees, longitudeInDegrees
	IFTTT	api.segment.io	anonymousId, app.namespace, app.version, device.advertisingId, device.id, device.manufacturer, device.model, device.name, locale, os.version, userId, login, timestamp
		e.crashlytics.com	appBundleId, appVersionName, deviceModel, osVersion, timestamp

Table 7. Trackers embedded in smartphone applications. *We see that Google is here also a major actor.*

Application	Trackers
Alexa	Amazon Advertisement
	Google CrashLytics
All4Hue	-
Home	Google Analytics
Home Assistant	n/a
Philips Hue	Amplitude
	Apptimize
	Braze
	Google CrashLytics
	Google Firebase Analytics
	HockeyApp
Hue Hello	Google CrashLytics
	Google Firebase Analytics
IFTTT	Google CrashLytics
	Google Firebase Analytics
	Segment
OpenHAB	Google Crashlytics
	Google Firebase Analytics
IKEA Tradfri	-

3.5 Features Beneficial to Privacy

Table 8 summarizes several privacy-related observations for the various techniques in the control part. The use of open-source solutions is beneficial to privacy because of the transparency it brings to the solution: personal data leaks can be identified and their adequacy in front of the processing analyzed, and if anyone feels there is something wrong, a privacy preserving fork can be easily created. From this point of view, the Home Assistant and OpenHAB open-source projects are the only two exceptions.

Then the following feature that one can expect from a privacy-preserving service is encryption. Among all solutions allowing a direct local connection to the Philips bridge or bulb, encryption is ignored by four solutions that entirely rely on the physical network security mechanisms. Even more importantly, it is essential that any message leaving the home network be encrypted. Fortunately, most solutions are using HTTPS connections because they rely on the external Web API servers hosted by Google or Amazon, for which encryption is mandatory. An exception is for the open-source solutions – Home Assistant and OpenHAB – that use HTTP by default, even if it is possible to switch to HTTPS.

However, as we have seen in Section 3.3, encryption alone is not enough as one can often infer what action is triggered by a frame, based on analyzing its size.

Table 8 reminds that only IFTTT and IKEA Tradfri are using fixed size messages to control the bulb, for which the user action (ON versus OFF) cannot be inferred.

4 Discussion

4.1 About the Ecosystem

We see that Google and Amazon have an overwhelming power in the smart home. Their scope goes from their own services to hosting the services of other manufacturers, but also selling smart speakers, smart devices (e.g., through NEST, owned by Alphabet just like Google), and even trackers for smartphone applications. The web giants are becoming the main authority selecting what

Table 8. Features beneficial to privacy. *The "open-source" column shows if the code of the application (smartphone), firmware (with smart speakers) or Raspberry Pi software, is publicly available. The two "encrypted communications" columns show if the application or device (incl. the Raspberry Pi when meaningful) use encrypted communications* **by default**, *respectively locally or remotely (when remote communications are not supported, we mark the entry "n/a"). The "fixed size messages" column shows if the application is sending fixed size control messages so that their meaning containing them cannot be inferred (Section 3.3). Finally, the "Works without Internet" column shows which control technique still works in case of Internet shortage, and as a side effect is less likely to leak personal data to remote servers.*

Control technique	open-source	encrypted local communications	encrypted remote communications	fixed size messages	works w/o Internet
Amazon Echo Spot	×	✓	✓	n/a	×
Google Home	×	✓	✓	n/a	×
IKEA Tradfri Remote Control	n/a	n/a	n/a	n/a	✓
Smartphone + Alexa app	×	✓	✓	×	×
Smartphone + All4Hue app	×	×	n/a	×	✓
Smartphone + Home app	×	✓	✓	×	×
Smartphone + R. Pi + Home Ass.	✓	×	×	×	✓
Smartphone + Philips Hue app	×	✓	✓	×	✓
Smartphone + Hue Hello app	×	×	n/a	×	✓
Smartphone + IFTTT app	×	✓	✓	✓	×
Smartphone + R. Pi + OpenHAB	✓	×	×	×	✓
Smartphone + IKEA Tradfri app + gw	×	✓	n/a	✓	✓

will be accessible online or not. They don't even make use of local servers in the country from where the service is accessed.

By looking at the captures made, it is also clear that data collection is their main goal: while switching ON or OFF a bulb remains a basic action, a lot of information is collected. It is also particularly concerning to see that Google or Amazon manufactured devices meant to listen to people at home, and didn't make it process as much data as possible locally instead of sending almost everything to their servers abroad. The 2019 Amazon scandal [24] doesn't make it less concerning. A project like Snips [25] shows that privacy preserving alternatives for smart assistants are feasible, achieving speech-to-text processing locally. Without a change in their business model, this mass collection is not likely to change.

An actor like IFTTT also has an unclear business model. It is supposed to be a free system with a business model based on providing tailored services to people willing to pay. But then why send everything to Internet remote servers when most of the requests could be done locally? Is this design motivated by a hidden data collection strategy?

4.2 Energetic Considerations

Finally, we think that the energy consumption question should be taken more seriously, as many IoT device manufacturers claim they are preventing energy waste but don't seem to actually do so. This topic is relatively new, and we couldn't find a lot of literature tackling those problems.

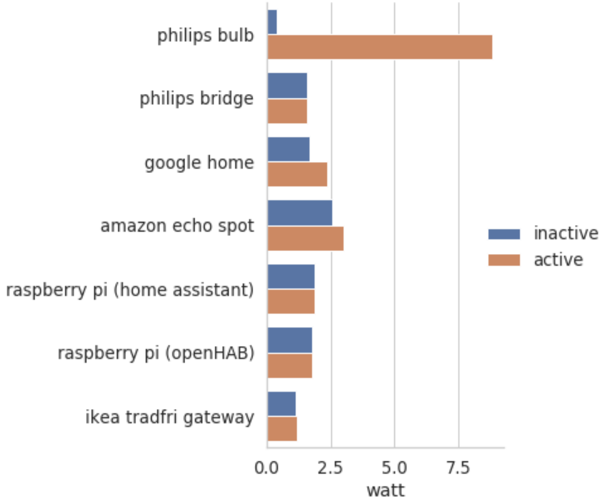


Fig. 6. Consumption of devices. *An active device is a device being used by the user. For the bulb, it means that it is ON and not in stand by mode.*

Figure 6 shows the energy consumption of our devices. If a normal bulb does not consume any energy when it is off, on the opposite, the smart bulb and the bridge are powered 24/7 and always consume energy, even in inactive state. The higher the number of smart bulbs in a house, the higher this recurrent power consumption, and the less the user uses them, the higher the difference compared to a standard bulb. For instance, let us assume the user owns one bulb switched on 2 hours a day. We measured that the Philips bulb and the normal bulb both consume 8.8 watts when they are on and that the smart bulb consumes 0.4 watts when inactive (0 for a standard bulb). We measured that the bridge consumes 1.6 watts continuously. It follows that, under these conditions, a standard bulb will consume $2 \times 8.8 = 17.6$ watt-hour daily while a smart bulb plus a bridge will consume $2 \times 8.8 + 22 \times 0.4 + 24 \times 1.6 = 64.8$ watt-hour. In those conditions, the smart bulb in those conditions consumes more than 3 times more energy than a normal one. Even if the result depends on the exact scenario, it shows that smart devices are not necessarily a panacea when it comes to energy consumption. Furthermore, this calculation totally ignores the energetic costs of data transmissions on the Internet, of cloud storage and cloud processing.

5 Related Work

The literature focuses most of the time on one device at a time, trying to find vulnerabilities, privacy leaks, or ways to secure transmissions. One well studied subject is device identification and wireless infrastructure reconnaissance [26] [27] [28] [29]. Then, once the devices in a network are discovered, research has been done on how to infer what actions are being triggered by those devices from packets metadata and cleartext frames [30] [31] [32]. Some research was also done on how to prevent inferences from network traffic [30] [33] [32]. Then there is the security aspect. Concerning Wi-Fi networks, there was some work done [27] on how to mitigate attacks by stopping attackers connections. Some research was done on how to detect a misbehaving smartphone application based on its source code and the ZigBee and Z-Wave traffic generated by the hub used by the application and a target device [34]. There are also attacks directly aimed at the security of connectivity technologies: Wi-Fi WPA2 [35], Bluetooth [36] and ZigBee [37].

Some tools exist to analyze [29] or act upon network traffic by implementing MITM attacks [38] or advertisement removal [39].

In contrast, our work focuses on different ways to control a given device, which is also paramount. Indeed, a device can sometimes be used by several other devices. To that extent, the controlled device will expose an API with different behaviours depending on the situation. Also the content of the encrypted traffic is often left out in related works, whereas we used several techniques to look into encrypted frames. We also focus more on the sovereignty aspect by looking at the data destinations and volumes, and the devices and smartphone applications design choices when it comes to privacy.

6 Conclusion

In this work, we studied how using different techniques to control a target device, here a Philips Hue smart bulb, can drastically change the security and privacy risks. We considered a broad range of such techniques: two smart speakers, a smart button that can directly control the bulb, seven smartphone applications – including that of the bulb manufacturer –, and two open source generic IoT device management systems.

Our work highlights several core issues. First of all, it highlights that a small change in the way one controls the target device may have major consequences from various viewpoints: security risks, privacy leaks, sovereignty, volume of traffic generated, ecological costs. It is not conceivable for a user to have to anticipate the consequences of choosing one technique over the other, since: (1) the natural tendency is to focus on the device itself, not on the control part of the device, whereas both are essential, and (2) there is no information really accessible as privacy policies, currently the only way to get information, either do not exist, are hard to find on the manufacturer's web site, are sometimes copied by other connected devices web sites which is confusing, are not accessible to the end-user's native language, are written by lawyers for lawyers, or are extremely imprecise. All of this is hidden behind easy-to-use user interfaces, that also contribute to fool the user into thinking he has the control, whereas the situation is just the opposite.

Secondly, it highlights the major role played by Google – and to a lesser extent Amazon – in the studied ecosystem. These actors are at the crossroad of many flows, thanks to their smart speakers, their own services (e.g., for Natural Language Processing), the services provided to other actors (e.g., Google provides its cloud services to Philips), and their application trackers. They are in position to know a lot about a smart home and its inhabitants life.

Finally, for many actors, our work highlights that there are design flaws, their solution being non compliant with privacy-by-design principles. Relying only on local communications, whenever appropriate (e.g., when the user is at home, using a device connected to Wi-Fi), should be the rule. Not only does this reduce the risks of personal data leaks, but it also makes the smart home robust in front of Internet impairment, it helps preventing sovereignty problems, it helps reducing the energy consumption of the smart home (e.g., no transatlantic communications nor cloud storage), in a context where the simple fact of having continuously powered devices to control a bulb makes the global energy consumption higher than that of a regular bulb. From this point of view, experiments (not shown) proved that filtering the periodic data communications from the Philips bridge to the Philips server (hosted by Google) did not prevent in any way to control the bulb.

"Data minimization" should also be the rule. We see no real justification in collecting many stable identifiers and other metadata in messages exchanged with remote servers by the IFTTT and Alexa applications. We see no real justification in having trackers in many smartphone applications.

From these two point of views, the IKEA Tradfri ecosystem, compatible with the Philips Hue smart bulb, is highly recommended, on the express condition that the user does not use the dedicated skill that enables a connection to Amazon Alexa or Google Assistant services, which would also ruin any privacy benefit. The Home Assistant and – to a lesser extent, given the presence of two trackers in the associated application – OpenHAB systems are also good alternatives that favor privacy over other considerations, while enabling remote control. The open-source nature of these two community projects also provides total transparency, a feature that severely lacks commercial products.

We therefore ask for different architectures and solutions since, with the exception of IKEA, Home Assistant and OpenHAB, the current situation is neither respectful of the user's privacy (many things happen without the user's informed consent), nor sovereign (it creates dependencies on foreign actors), nor sustainable (it has a significant ecological cost). It is also obvious that the situation is too complex for the end-user to be in position to clearly understand the privacy implications of the various choices available to him to control his smart bulb. This is also a major concern in case of a European user, since the GDPR requires a clear and informed consent before a data controller can collect and process any personal data.

7 Acknowledgment

This work has been supported by the DAPCODS/IOTics ANR 2016 project (ANR-16-CE25-0015).

The authors also want to thank Nataliia Bielova and Piyush Patil for their comments and help in this work.

References

[1] "Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016." [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016

[2] "2018 reform of EU data protection rules." [Online]. Available: https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules_en

[3] "Buy the Philips Hue White Single bulb E26 046677530341 Single bulb E26." [Online]. Available: https://www2.meethue.com/en-us/p/hue-white-single-bulb-e26/046677530341

[4] "Echo Spot | Alexa-enabled Speaker with 2.5" Screen - Black." [Online]. Available: https://www.amazon.com/Echo-Spot-Smart-Display-Alexa/dp/B073SQYXTW

[5] "Google Home." [Online]. Available: https://store.google.com/product/google_home

[6] "Gateway TRÅDFRI - IKEA." [Online]. Available: https://www.ikea.com/gb/en/p/tradfri-gateway-white-20337807/

[7] "Remote control TRÅDFRI - IKEA." [Online]. Available: https://www.ikea.com/gb/en/p/tradfri-remote-control-30338849/

[8] "LG Nexus 5 - Full phone specifications." [Online]. Available: https://www.gsmarena.com/lg_nexus_5-5705.php

[9] "Buy the Philips Hue Bridge 046677458478 Bridge." [Online]. Available: https://www2.meethue.com/en-us/p/hue-bridge/046677458478

[10] "Raspberry Pi 3 Model B." [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[11] "Amazon Alexa – Applications sur Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=com.amazon.dee.app&hl=fr

[12] "all 4 hue – Applications sur Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=de.renewahl.all4hue&hl=fr

[13] "Google Home – Applications sur Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=com.google.android.apps.chromecast.app&hl=fr

[14] "Chrome : rapide et sécurisé – Applications sur Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=com.android.chrome&hl=fr

[15] "Philips Hue – Applications sur Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=com.philips.lighting.hue2&hl=fr

[16] "Hue Hello (For Philips Hue Lights) – Applications sur Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=com.prodpeak.huehello&hl=fr

[17] "IFTTT – Applications sur Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=com.ifttt.ifttt&hl=fr

[18] "openHAB – Applications sur Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=org.openhab.habdroid&hl=fr

[19] "IKEA Home smart (TRÅDFRI) – Applications sur Google Play." [Online]. Available: https://play.google.com/store/apps/details?id=com.ikea.tradfri.lighting

[20] "Python Data Analysis Library — pandas: Python Data Analysis Library." [Online]. Available: https://pandas.pydata.org/

[21] "Burp Suite Scanner | PortSwigger." [Online]. Available: https://portswigger.net/burp

[22] P. Cipolloni, "Universal Android SSL Pinning bypass with Frida | @Mediaservice.net Technical Blog." [Online]. Available: https://techblog.mediaservice.net/2017/07/universal-android-ssl-pinning-bypass-with-frida/

[23] "Exodus Privacy." [Online]. Available: http://exodus-privacy.eu.org/en/

[24] L. H. Newman, "How To Tighten Your Amazon Echo and Google Home Privacy," *Wired*, Apr. 2019. [Online]. Available: https://www.wired.com/story/alexa-google-assistant-echo-smart-speaker-privacy-controls/

[25] "Snips - Using Voice to Make Technology Disappear." [Online]. Available: https://snips.ai/

[26] S. Siby, R. R. Maiti, and N. Tippenhauer, "IoTScanner: Detecting and Classifying Privacy Threats in IoT Neighborhoods," *arXiv preprint arXiv:1701.05007*, 2017. [Online]. Available: https://arxiv.org/abs/1701.05007

[27] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. R. Sadeghi, and S. Tarkoma, "IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, Jun. 2017, pp. 2177–2184.

[28] "Fast Web-based Attacks to Discover and Control IoT Devices." [Online]. Available: https://freedom-to-tinker.com/2018/06/21/fast-web-based-attacks-to-discover-and-control-iot-devices/

[29] "Security Analysis for IoT devices | Completely Automated." [Online]. Available: https://www.iot-inspector.com/

[30] N. Apthorpe, D. Reisman, and N. Feamster, "A Smart Home is No Castle: Privacy Vulnerabilities of Encrypted IoT Traffic," *arXiv:1705.06805 [cs]*, May 2017, arXiv: 1705.06805. [Online]. Available: http://arxiv.org/abs/1705.06805

[31] D. Wood, N. Apthorpe, and N. Feamster, "Cleartext Data Transmissions in Consumer IoT Medical Devices," in *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*. ACM, 2017, pp. 7–12.

[32] N. Apthorpe, D. Y. Huang, D. Reisman, A. Narayanan, and N. Feamster, "Keeping the Smart Home Private with Smart(er) IoT Traffic Shaping," *arXiv:1812.00955 [cs]*, Dec. 2018, arXiv: 1812.00955. [Online]. Available: http://arxiv.org/abs/1812.00955

[33] N. Apthorpe, D. Reisman, and N. Feamster, "Closing the Blinds: Four Strategies for Protecting Smart Home Privacy from Network Observers," *arXiv preprint arXiv:1705.06809*, 2017.

[34] W. Zhang, Y. Meng, Y. Liu, X. Zhang, Y. Zhang, and H. Zhu, "HoMonit: Monitoring Smart Home Apps from Encrypted Traffic," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: ACM, 2018, pp. 1074–1088, event-place: Toronto, Canada. [Online]. Available: http://doi.acm.org/10.1145/3243734.3243820

[35] M. Vanhoef and F. Piessens, "Key reinstallation attacks: Forcing nonce reuse in WPA2," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1313–1328.

[36] M. Ryan, "Bluetooth: With Low Energy Comes Low Security." *WOOT*, vol. 13, pp. 4–4, 2013.

[37] X. Cao, D. M. Shila, Y. Cheng, Z. Yang, Y. Zhou, and J. Chen, "Ghost-in-ZigBee: Energy Depletion Attack on ZigBee-Based Wireless Networks," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 816–829, Oct. 2016.

[38] "PiRogue is a small device meant to ease network interception and analysis.: PiRanhaLysis/PiRogue," Mar. 2019, original-date: 2018-03-11T12:10:17Z. [Online]. Available: https://github.com/PiRanhaLysis/PiRogue

[39] "Pi-hole®: A black hole for Internet advertisements." [Online]. Available: https://pi-hole.net/