Privacy challenges in wireless communications of the Internet of Things

Guillaume Celosia

HAL Id: tel-02974989

https://theses.hal.science/tel-02974989v2

Submitted on 7 Jan 2021

N° d'ordre NNT : 2020LYSEI069

THÈSE DE DOCTORAT DE L'UNIVERSITÉ DE LYON
opérée au sein de
INSA Lyon

École Doctorale ED 512
InfoMaths

Spécialité/discipline de doctorat :
Informatique

Soutenue publiquement le 22/09/2020, par :
Guillaume CELOSIA

Privacy Challenges in Wireless Communications of the Internet of Things

Devant le jury composé de :

Kasper RASMUSSEN	Associate Professor	Univ. of Oxford	Rapporteur
Bernard TOURANCHEAU	Professeur des Universités	Univ. Grenoble Alpes	Rapporteur
Sonia BEN MOKHTAR	Directeur de Recherche	CNRS	Examinateur
Jean-Marie GORCE	Professeur des Universités	INSA Lyon	Examinateur
Vincent NICOMETTE	Professeur des Universités	INSA Toulouse	Examinateur
Valérie VIET TRIEM TONG	Professeur	CentraleSupélec	Examinateur
Daniel LE MÉTAYER	Directeur de Recherche	Inria	Directeur de thèse
Mathieu CUNCHE	Maître de Conférences	INSA Lyon	co Directeur de thèse

Cette thèse a été préparée à :

Laboratoire CITI (Centre d'Innovation en Télécommunications et Intégration de services) – INSA Lyon
Bâtiment Claude Chappe
6 Avenue des Arts
69100 Villeurbanne
France

dans le cadre de :

Équipe Inria PRIVATICS
Centre Inria Grenoble – Rhône-Alpes
Inria – Antenne Lyon – La Doua
Bâtiment CEI-2
56 Boulevard Niels Bohr
69100 Villeurbanne
France

sur un financement de :

Chaire IoT INSA Lyon – SPIE ICS

Département FEDORA – INSA Lyon – Écoles Doctorales – Quinquennal 2016-2020

SIGLE	ÉCOLE DOCTORALE	NOM ET COORDONNÉES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **http://edchimie-lyon.fr** Sec. : Renée EL MELHEM Bâtiment Blaise Pascal – 3e étage INSA : R. GOURDON secretariat@edchimie-lyon.fr	M. Stéphane DANIELE Institut de Recherches sur la Catalyse et l'Environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 Avenue Albert Einstein 69100 Villeurbanne directeur@edchimie-lyon.fr
E.E.A.	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** **http://edeea.ec-lyon.fr** Sec. : M.C. HAVGOUDOUKIAN ecole-doctorale.eea@ec-lyon.fr	M. Gérard SCORLETTI École Centrale de Lyon 36 Avenue Guy de Collongue 69130 Ecully Tél. : 04 72 18 60 97, Fax : 04 78 43 37 17 gerard.scorletti@ec-lyon.fr
E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** **http://e2m2.universite-lyon.fr** Sec. : Sylvie ROBERJOT Bâtiment Atrium, UCB Lyon 1 Tél. : 04 72 44 83 62 INSA : H. CHARLES secretariat.e2m2@univ-lyon1.fr	M. Philippe NORMAND UMR 5557 Lab. d'Écologie Microbienne Université Claude Bernard Lyon 1 Bâtiment Mendel 43 Boulevard du 11 Novembre 1918 69100 Villeurbanne philippe.normand@univ-lyon1.fr
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** **http://ediss-lyon.fr** Sec. : Sylvie ROBERJOT Bâtiment Atrium, UCB Lyon 1 Tél. : 04 72 44 83 62 INSA : M. LAGARDE secretariat.ediss@univ-lyon1.fr	Mme Sylvie RICARD-BLUM Institut de Chimie et Biochimie Moléculaires et Supramoléculaires ICBMS-UMR 5246 CNRS, Univ. Lyon 1 Bâtiment Curien – 3e étage Nord 43 Boulevard du 11 Novembre 1918 69100 Villeurbanne Tél. : 04 72 44 82 32 sylvie.ricard-blum@univ-lyon1.fr
INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** **http://edinfomaths.universite-lyon.fr** Sec. : Renée EL MELHEM Bâtiment Blaise Pascal – 3e étage Tél. : 04 72 43 80 46 infomaths@univ-lyon1.fr	M. Hamamache KHEDDOUCI Bâtiment Nautibus 43 Boulevard du 11 Novembre 1918 69100 Villeurbanne Tél. : 04 72 44 83 69 hamamache.kheddouci@univ-lyon1.fr
Matériaux	**MATÉRIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec. : Stéphanie CAUVIN Bâtiment Direction Tél. : 04 72 43 71 70 ed.materiaux@insa-lyon.fr	M. Jean-Yves BUFFIÈRE INSA Lyon MATÉIS Bâtiment Saint-Exupéry 7 Avenue Jean Capelle 69100 Villeurbanne Tél. : 04 72 43 71 70, Fax 04 72 43 85 28 jean-yves.buffiere@insa-lyon.fr
MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** **http://edmega.universite-lyon.fr** Sec. : Stéphanie CAUVIN Bâtiment Direction Tél. : 04 72 43 71 70 mega@insa-lyon.fr	M. Jocelyn BONJOUR INSA Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9 Rue de la Physique 69100 Villeurbanne jocelyn.bonjour@insa-lyon.fr
ScSo	**ScSo*** **http://ed483.univ-lyon2.fr** Sec. : Véronique GUICHARD Tél. : 04 78 69 72 76 INSA : J.Y. TOUSSAINT veronique.cervantes@univ-lyon2.fr	M. Christian MONTES Université Lyon 2 86 Rue Pasteur 69007 Lyon christian.montes@univ-lyon2.fr

* ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie.

Remerciements

La réalisation de cette thèse a été possible uniquement grâce au concours de plusieurs personnes auxquelles je souhaite adresser mes remerciements, mon respect, et ma gratitude. En particulier...

...à mon encadrant, Mathieu Cunche, pour ta confiance, ainsi que pour le travail scientifique et administratif fourni;

...à mon directeur de thèse, Daniel Le Métayer, pour tes conseils avisés;

...aux assistantes d'équipe, Helen Pouchot, Aurélie Reymond, et Linda Soumari, pour la facilitation des démarches administratives, aussi nombreuses et interminables furent-elles;

...à Célestin Matte, à Alexis Duque, au groupe de recherche Furious MAC de la *United States Naval Academy*, ainsi qu'aux examinateurs de la conférence PETS, pour leurs idées, pour le partage d'un corpus de traces de trafic réseau (voir Section III.5.4.1), et pour leurs commentaires perspicaces en vue d'améliorer les articles scientifiques publiés;

...à mon Arnestre [1], pour tes bons petits plats, et pour les parties de rigolade que nous avons partagé ensemble. Merci également d'avoir fait vivre en moi cette passion pour l'informatique, pour les valeurs de la vie que tu m'as inculqué, ainsi que pour l'éducation exemplaire que j'ai eu la chance de recevoir de ta part;

...à mon Simi [1], pour ces soirées films endiablées, et pour ces innombrables maux que tu as su soigner de tes mains d'or. Merci pour nos discussions, pour cet idéal que tu représentes quotidiennement et vers lequel j'ambitionne à tendre, ainsi que pour avoir façonné de ta bienveillance la personne que je suis aujourd'hui;

...à mon Didodu [1], pour ces années passées à tes côtés, pour ces sessions de jeux vidéos déjantées, et pour cette complicité que nous avons su entretenir pendant plus de vingt-deux ans. Merci d'être ce frère extraordinaire, qui nourrit ma culture sur des sujets divers et variés, et qui fleurit mon chemin de vie de par sa simple présence;

...à ma Memel [1], pour ton amour, pour ton soutien permanent, et pour nos longs débats métaphysiques qui réchauffent mon cœur. Merci pour toutes ces fois où tu as su trouver les mots justes afin de me permettre d'avancer significativement dans mes recherches. Tout au long de cette thèse tu as été mon atout majeur, et je t'en serai éternellement reconnaissant.

---

1. Cette thèse est l'aboutissement d'une longue période durant laquelle vous n'avez jamais cessé de croire en moi. Ce manuscrit vous est intégralement dédié. *"Unis nous sommes, unis nous avons vaincu, unis nous vaincrons"*.

“Moi, l'informatique, je n'y comprends ria.“
Arnestre de Saint-Barthre

Résumé

Également connue sous le nom d'*Internet des Objets* (IdO), la prolifération des objets connectés offre des opportunités sans précédent aux consommateurs. Des moniteurs d'activité physique aux assistants médicaux, en passant par les appareils électroménagers pour maisons intelligentes, les objets IdO évoluent dans une pléthore de domaines d'application.

Cependant, les avantages qu'ils peuvent apporter à notre société augmentent conjointement avec leurs implications en matière de vie privée. Communiquant continuellement de précieuses informations par le biais de liaisons non filaires telles que le Bluetooth et le Wi-Fi, ces appareils connectés accompagnent leurs propriétaires dans leurs activités. La plupart du temps émises sur des canaux ouverts, et parfois en l'absence de chiffrement, ces informations sont alors facilement accessibles pour tout attaquant passif à portée.

Dans cette thèse, nous explorons deux problèmes de vie privée majeurs résultant de l'expansion de l'IdO et de ses communications sans fil : le *traçage physique* et l'*inférence d'informations utilisateurs*. Sur la base de deux grands ensembles de données composés de signaux radio issus de périphériques Bluetooth/BLE, nous mettons d'abord en échec les fonctionnalités anti-traçage existantes avant de détailler plusieurs applications invasives pour la vie privée. En s'appuyant sur des attaques passives et actives, nous démontrons également que les messages diffusés contiennent des informations en clair allant des caractéristiques techniques des appareils aux données personnelles des utilisateurs telles que des adresses e-mail et numéros de téléphone.

Dans un second temps, nous concevons des contre-mesures pratiques pour résoudre les problèmes de vie privée identifiés. Dans ce sens, nous fournissons des recommandations aux fabricants, et proposons une approche afin de vérifier l'absence de failles dans l'implémentation de leurs protocoles.

Enfin, dans le but d'illustrer davantage les menaces de vie privée enquêtées, nous implémentons deux démonstrateurs. Par conséquent, *Venom* introduit un système de traçage physique visuel et expérimental, tandis qu'*Himiko* propose une interface humaine permettant d'inférer des informations sur les appareils IdO et leurs propriétaires.

Mots-clés : Vie privée; Réseaux sans fil; Internet des Objets; Objets connectés; Traçage physique; Inférence d'informations utilisateurs; Bluetooth; BLE.

Abstract

Also known as the *Internet of Things* (IoT), the proliferation of connected objects offers unprecedented opportunities to consumers. From fitness trackers to medical assistants, through smarthome appliances, the IoT objects are evolving in a plethora of application fields.

However, the benefits that they can bring to our society increase along with their privacy implications. Continuously communicating valuable information via wireless links such as Bluetooth and Wi-Fi, those connected devices support their owners within their activities. Most of the time emitted on open channels, and sometimes in the absence of encryption, those information are then easily accessible to any passive attacker in range.

In this thesis, we explore two major privacy concerns resulting from the expansion of the IoT and its wireless communications: *physical tracking* and *inference of users information*. Based on two large datasets composed of radio signals from Bluetooth/BLE devices, we first defeat existing anti-tracking features prior to detail several privacy invasive applications. Relying on passive and active attacks, we also demonstrate that broadcasted messages contain cleartext information ranging from the devices technical characteristics to personal data of the users such as e-mail addresses and phone numbers.

In a second time, we design practical countermeasures to address the identified privacy issues. In this direction, we provide recommendations to manufacturers, and propose an approach to verify the absence of flaws in the implementation of their protocols.

Finally, to further illustrate the investigated privacy threats, we implement two demonstrators. As a result, *Venom* introduces a visual and experimental physical tracking system, while *Himiko* proposes a human interface allowing to infer information on IoT devices and their owners.

Keywords : Privacy; Wireless networks; Internet of Things; Connected objects; Physical tracking; Inference of users information; Bluetooth; BLE.

Contents

List of Figures

List of Tables

List of Acronyms

1xRTT (1x) Radio Transmission Technology

AD Advertising Data

AdvA Advertising Address

AES Advanced Encryption Standard

API Application Programming Interfaces

AR Activity Recognition

ATT Attribute

AUC Area Under the Curve

BD_ADDR Bluetooth Device Address

BLE Bluetooth Low Energy

BT Bluetooth

CDP Connected Devices Platform

CM Continuity Messages

CNIL Commission Nationale de l'Informatique et des Libertés

COVID-19 Corona Virus Disease 2019

CPU Central Processing Unit

CRC Cyclic Redundancy Check

CSR Cambridge Silicon Radio

CSS Cascading Style Sheets

D2D Device-to-Device

DPA Data Protection Authorities

DRM Digital Rights Management

DSID Destination Signaling Identifier

ECB Electronic Code Book

EPIC Electronic Privacy Information Center

ePR ePrivacy Regulation

EU European Union

EV-DO Evolution-Data Optimized

FFT Fast Fourier Transform

FIPS Federal Information Processing Standards

FPR False Positive Rate

GATT Generic Attribute

GCM Galois/Counter Mode

GDPR General Data Protection Regulation

GHz Gigahertz

GNU GNU's Not Unix

GPL General Public License

GPRS General Packet Radio Service

GPS Global Positioning System

GSM Global System for Mobile communications

GSN Global State Number

H/s Hashes per second

HAP HomeKit Accessory Protocol

HEC Header Error Check

IAB Interactive Advertising Bureau

ICT Information and Communications Technology

ID Identifier

IdO Internet des Objets

IE Information Element

IEEE Institute of Electrical and Electronics Engineers

IETF Internet Engineering Task Force

IFTTT If This Then That

INSA Institut National des Sciences Appliquées

INSEE Institut National de la Statistique et des Etudes Economiques

IoT Internet of Things

IP Internet Protocol

IRK Identity Resolving Key

ISM Industrial, Scientific and Medical

IV Initialization Vector

JSON JavaScript Object Notation

KS Kolmogorov-Smirnov

L2CAP Logical Link Control and Adaptation Protocol

LAP Lower Address Part

LE Low Energy

LINC Laboratoire d'Innovation Numérique de la CNIL

LPD433 Low Power Device 433 MHz

LPWAN Low Power Wide Area Network

LSB Least Significant Bit

LTE Long Term Evolution

LTS Long Term Support

M2M Machine-to-Machine

MAC Media Access Control

MEDS Medical Data Service

MHz Megahertz

MLS Mozilla Location Service

MS Media Service

MSB Most Significant Bit

NAP Non-significant Address Part

NCS Notification Center Service

NFC Near Field Communication

NIC Network Interface Controller

OS Operating System

OSI Open Systems Interconnection

OUI Organizationally Unique Identifier

PAN Personal Area Network

PDU Protocol Data Unit

PETS Privacy Enhancing Technologies Symposium

PHP PHP: Hypertext Preprocessor

PIA Privacy Impact Assessment

PII Personally Identifiable Information

PnP Plug and Play

PRH Philips Robust Hash

PRNG Pseudorandom Number Generator

RFC Request For Comments

RFID Radio-Frequency Identification

ROC Receiver Operating Characteristic

RSSI Received Signal Strength Indication

RTT Round-Trip Time

SHA Secure Hash Algorithm

SIG Special Interest Group

SNR Signal-to-Noise Ratio

SSID Service Set Identifier

TLV Type-Length-Value

TPMS Tire Pressure Monitoring System

TPR True Positive Rate

TV Television

UAP Upper Address Part

UDHR Universal Declaration of Human Rights

UHF Ultra High Frequency

USB Universal Serial Bus

UTF Universal Character Set Transformation Format

UUID Universally Unique Identifier

W3C World Wide Web Consortium

WAN Wide Area Network

WMBUS Wireless Meter Bus

WPAN Wireless Personal Area Network

WPS Wi-Fi Protected Setup

List of Contributions

From our work, emerging results have been disseminated in some top ranking conferences for privacy and security. Similarly, we implemented and released several software under an open-source license (GNU General Public License (GPL) v3.0) in order to promote their wide diffusion. Listed as follows, publications, presentations and produced software are sorted from the most recent to the oldest.

Journal Articles

1. Guillaume Celosia and Mathieu Cunche. Discontinued Privacy: Personal Data Leaks in Apple Bluetooth-Low-Energy Continuity Protocols. *Proceedings on Privacy Enhancing Technologies*, 2020(1):26–46, 2020

International Conferences

1. Guillaume Celosia and Mathieu Cunche. Valkyrie: A Generic Framework for Verifying Privacy Provisions in Wireless Networks. In *Proceedings of the 13th Conference on Security and Privacy in Wireless and Mobile Networks*, pages 278–283, 2020

2. Guillaume Celosia and Mathieu Cunche. DEMO: Venom: a Visual and Experimental Bluetooth Low Energy Tracking System. In *Proceedings of the 13th Conference on Security and Privacy in Wireless and Mobile Networks*, pages 346–348, 2020

3. Guillaume Celosia and Mathieu Cunche. Saving Private Addresses: An Analysis of Privacy Issues in the Bluetooth-Low-Energy Advertising Mechanism. In *MobiQuitous 2019 – 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 444–453, 2019

4. Guillaume Celosia and Mathieu Cunche. DEMO: Himiko: A Human Interface for Monitoring and Inferring Knowledge on Bluetooth-Low-Energy Objects. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, pages 292–293, 2019

5. Guillaume Celosia and Mathieu Cunche. Detecting smartphone state changes through a Bluetooth based timing attack. In *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 154–159, 2018

International Workshops

1. Guillaume Celosia and Mathieu Cunche. Fingerprinting Bluetooth-Low-Energy Devices Based on the Generic Attribute Profile. In *Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things*, pages 24–31, 2019

National [1] Conferences/Workshops

1. Guillaume Celosia and Mathieu Cunche. Saving Private Addresses: An Analysis of Privacy Issues in the Bluetooth-Low-Energy Advertising Mechanism. In *Atelier sur la Protection de la Vie Privée (APVP)*, 2019

2. Guillaume Celosia and Mathieu Cunche. Detecting smartphone state changes through a Bluetooth based timing attack. In *Atelier sur la Protection de la Vie Privée (APVP)*, 2018

3. Guillaume Celosia and Mathieu Cunche. Protection de la Vie Privée dans les Communications Sans Fil de l'Internet des Objets (IoT). In *Rendez-vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information (RESSI)*, 2018

4. Guillaume Celosia and Mathieu Cunche. POSTER: Preserving Privacy in Wireless Communications of the Internet of Things (IoT). In *Rendez-vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information (RESSI)*, 2018

Additional presentations

1. Guillaume Celosia and Mathieu Cunche. Discontinued Privacy: Personal Data Leaks in Apple Bluetooth-Low-Energy Continuity Protocols. In *Annual seminar of the Inria PRIVATICS team*, 2020

2. Guillaume Celosia and Mathieu Cunche. DEMO: Himiko: A Human Interface for Monitoring and Inferring Knowledge on Bluetooth-Low-Energy Objects. In *Salon de l'Internet Des Objets (SIDO)*, 2019

3. Guillaume Celosia and Mathieu Cunche. Privacy Threats from the Bluetooth Low Energy Service Discovery Mechanism. In *Annual seminar of the Inria PRIVATICS team*, 2019

4. Guillaume Celosia and Mathieu Cunche. Detection of Bluetooth surveillance systems. In *Presentation for Inria EPI: Data and Algorithmic Transparency and Accountability (DATA)*, 2018

5. Guillaume Celosia and Mathieu Cunche. Privacy concerns in IoT radio communications. In *Pitch for the INSA Lyon – SPIE ICS IoT chair*, 2018

6. Mathieu Cunche, Célestin Matte, and Guillaume Celosia. Wi-Fi Scanner: How many data can be collected on your smartphone ? In *The Web Conference*, 2018

---

1. In France.

7. Guillaume Celosia and Mathieu Cunche. Preserving Privacy in Wireless Communications of the Internet of Things (IoT). In *Annual seminar of the Inria PRIVATICS team*, 2018

Software production

1. *Valkyrie* [21], a generic framework for verifying privacy provisions in wireless networks

2. *Joker* [22], a `Wireshark` dissector for BLE advertisement packets of *Apple* Continuity, *Microsoft* CDP and *Garmin* proprietary protocols

3. *Venom*, an experimental BLE tracking system (not public yet)

4. *Himiko*, a human interface for monitoring and inferring knowledge on BLE objects (not public yet)

5. *Koba*, a Python library to parse BLE advertising data (not public yet)

Chapter I

Introduction

This chapter introduces the privacy challenges induced by the increasing adoption of wireless communicating devices. To glimpse the purpose of our study, the Internet of Things (IoT) concept as well as its privacy implications are clarified in Section I.1. Afterwards, Section I.2 provides technical details on the radio communication standards considered throughout this thesis. Synthesizing its content, the structure of this manuscript is developed in Section I.3. To finish, Section I.4 furnishes a brief notice delimiting the efforts and contributions of each researcher who participated in this work.

I.1   Internet of Things and privacy

The *Internet of Things* (IoT) is a concept introduced to designate connected objects that form a network of physical devices. Emerging from overlapping trends such as the miniaturization, widespread network access and location positioning technology, the number of IoT devices is growing quickly. As of today, those connected objects are estimated to 31 billion units, and it is predicted that they will reach 75 billion in 2025 [23].

Generally equipped with sensors and/or actuators, IoT devices share the ability to sense, analyze their environments and communicate. To this end, they rely on autonomous communication between physical objects within the Internet infrastructure. Therefore, it is possible to augment those objects with knowledge of the Internet in various fields of application such as transport, manufacturing and agriculture, for instance.

With the wide and rapid adoption of the IoT, the number of applications in which users are involved as well as the level of interaction with the user are both bound to increase. As an illustration, temperatures and lights of a smarthome adapt to inhabitants, while smart meters collect and report their energy consumption; in healthcare, medical sensors and implants check the health of the patients and provide medical assistance; in fitness, applications on wristbands record and monitor physical activities of sportsmen; etc.

Overall, the IoT redefines the digital frontiers impacting our society on the economic, technical, social, industrial and political levels [24]. Moreover, focusing on the ameliorations brought by the deployment of such connected devices, the IoT can be beneficial for humanity. As an example, it has the potential to improve road safety, children care, health related analysis, entertainment and even the knowledge of ourselves through the quantified self.

To be efficient, the IoT applications require ever more sensors to be set up into our environments. From microphones to video cameras, through thermal and motion sensors, the establishment of such a diverse and broad sensor fabric into our society induces risks that have to be investigated.

As one of the most discussed risk, the privacy protection of users represents an important obstacle to the IoT adoption [25]. From one side, privacy is assimilated as a problem that is not relevant in our current society as it goes against the constant exhibition of our daily lives through social networks [26]. From the other side, privacy is believed not to be necessary *if you have nothing to hide* [27].

Within this heterogeneous situation, legislators are working to establish a legal framework for privacy. For instance, the European Parliament seriously takes into account this problem along with its associated debates to legislate on those questions [28]. Note that, specified as a fundamental right for everyone in the Universal Declaration of Human Rights (UDHR) [29], the privacy protection is still one of the discussion subjects in the European Parliament.

However, privacy remains difficult to define and formalize because of the complexity of the contexts that have to be considered. To glimpse this complexity, privacy is first defined in 1890 as *"the right to be left alone"* [30], while Westin [31] later considers that privacy *"provides individuals and groups in society with a preservation of autonomy, a release from role-playing, a time for self-evaluation and for protected communication"*. In turn, the UDHR stipulates that *"no one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honour and reputation. Everyone has the right to the protection of the law against such interference or attacks"* [29, Article 12]. As a result, whether it is focused on what happens to individuals when their information are used [30, 29] or on access and exposition of users information [31], there exist essential differences between those definitions.

In the meantime, behaviors, preferences, and activities of consumers are being more regularly recorded and monitored with the rise of the IoT. Indeed, connected objects are collecting data on users prior to send them to remote servers in order to enable a variety of services.

As a consequence, privacy have to be ensured within devices, during the storage, transmission and processing of users information [32]. In this direction, we can cite works done on data protection and privacy in IoT [33, 34], but also on the formalization of privacy policies and preferences to protect consumers [35, 36, 37].

In those researches, we noticed the difficulty that users can have to manually define their own privacy policies depending on each context. Actually, they are often ill-prepared to deal with the complexity of the systems, and thus do not manage to conceive the impact of sharing their data on their privacy. During our study, we faced supplementary privacy related ambiguities such as the lack of information on how users information are stored and the lack of guidelines plainly stating how to delete those data. As there is a clear momentum in the development of the IoT, companies have to better explain their practices regarding the processing of personal information.

In this thesis, we explore privacy challenges that wireless communications of the IoT devices can raise. More precisely, we focus our work on the data exposed by connected objects to *physically track* and *infer information* on their corresponding owners.

I.2 Background

In this section, we furnish technical details on both the Bluetooth and Bluetooth Low Energy (BLE) radio communication standards to understand our research. In this regard, we underline that we do not provide a complete overview of such standards. Nevertheless, we cover particular points that are further exploited during our studies.

I.2.1 Bluetooth

Bluetooth operates within the 2.4 GHz Industrial, Scientific and Medical (ISM) band through 79 1-MHz channels. To exchange data over radio communications, this technology leans on a multi-layered stack split into two parts [38, Vol 4, Part E, sec. 1.1] (see Figure I.1).

Implemented in microprocessors dongles, the *Controller* stack contains the Bluetooth radio interface, while the *Host* is an operating system part that processes high level data. Actually, the Logical Link Control and Adaptation Protocol (L2CAP) layer provides a link between those two stacks. Moreover, this packet based protocol follows channels communication models where specific commands can be sent between devices.

As defined in the Bluetooth Core Specification [38, Vol 3, Part A, sec. 4.8], ping relies on a request-response mechanism of the L2CAP layer: an `Echo Request` command is sent to a remote L2CAP device, and solicits an `Echo Response` back. As those requests are only designed to test a link between two Bluetooth devices, it is important to mention that they do not require any mutual authentication.
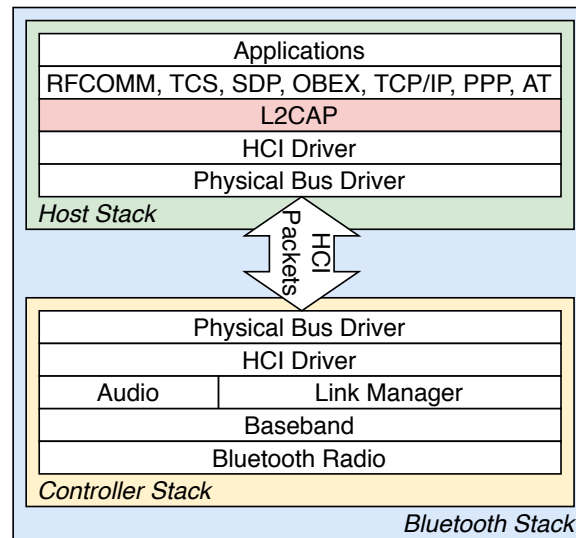
Figure I.1 – Representation of the Bluetooth stack along with its Logical Link Control and Adaptation Protocol (L2CAP) layer.
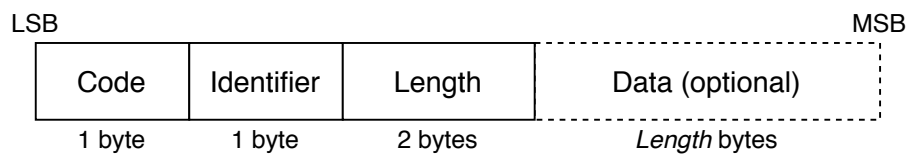
Figure I.2 – Format of the L2CAP `Echo` packet.

As presented in Figure I.2, the `Echo Request` and `Echo Response` packets share the same format and include the following fields:

— A Bluetooth device address (BD_ADDR) (48 bits): the destination device address;
— `Code` (8 bits): identifies the command type (`0x08` and `0x09` respectively specify `Echo Request` and `Echo Response`);
— `Identifier` (8 bits): used to match `Echo Request` with `Echo Response`;
— `Length` (16 bits): represents the length of the `Data` field;
— `Data`: contains optional and implementation specific data to be sent. If present, L2CAP entities shall ignore the content of this field [38, Vol 3, Part A, sec. 4.8].

In addition, the `Identifier`, `Length` and `Data` fields of the `Echo Request` are copied to the `Echo Response` during the request-response exchanges.

As a manufacturer unique device identifier [38, Vol 2, Part B, sec. 1.2], the BD_ADDR is divided into three parts: the Non-significant Address Part (NAP), the Upper Address Part (UAP) and the Lower Address Part (LAP), which are respectively the 16 most significant bits (MSB), the following 8 bits and the 24 least significant bits (LSB) of the BD_ADDR (see Figure I.3).
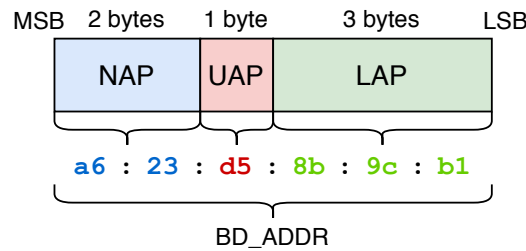
Figure I.3 – Format of the Bluetooth device address (BD_ADDR) including the Non-significant Address Part (NAP), the Upper Address Part (UAP) and the Lower Address Part (LAP).

Note that, the (UAP, LAP) pair is the minimum needed to establish a link with a remote device.

I.2.2   Bluetooth Low Energy (BLE)

Bluetooth Low Energy (BLE) is a lightweight subset of Bluetooth that has been introduced in 2010 as part of the Bluetooth Core Specification version 4.0 [39, Vol 6]. In fact, BLE and Bluetooth share elements such as range and radio frequency band[1] but differ on several points such as channel mapping, data transfer rate and power consumption.

Indeed, although Bluetooth can handle a lot of data at the cost of a high energy consumption, BLE is plebiscited by battery-powered appliances that only need to periodically exchange small amounts of data.

As a consequence, the BLE technology is more suitable for connected objects with low power computing resources which need to synchronize data with their applications. For instance, heart rate monitors and smartwatches rely on BLE to communicate with nearby smartphones and tablets.

I.2.2.1   Protocol

Instead of the Bluetooth 79 1-MHz channels, BLE operates on 40 2-MHz physical channels. Three of those channels – channel 37 (2402 MHz), 38 (2426 MHz) and 39 (2480 MHz) – are dedicated to the discovery mechanism called *advertisement*, while the 37 remaining ones are leveraged for data transmissions [38, Vol 6, Part B, sec. 1.4] (see Figure I.4).

As a result, BLE communications can be of two types: *broadcasted* and *connected*. Compared to connection, we highlight that the broadcasting provides neither security nor privacy protection as BLE devices in range are able to receive data emitted in clear over advertisement packets.

---

1. As for Bluetooth, BLE is a 2.4 GHz ISM band radio communication standard.
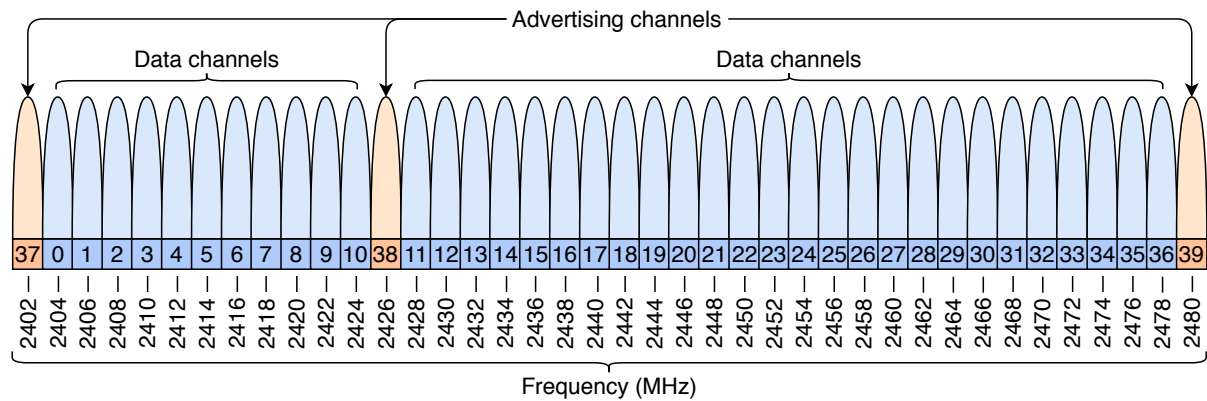
Figure I.4 – Representation of the Bluetooth Low Energy (BLE) spectrum emphasizing the advertising channels (channel 37 (2402 MHz), 38 (2426 MHz) and 39 (2480 MHz)).

Depending on the situation, a BLE device can endorse various roles following a Client-Server scheme. In order to simplify our explanations, we solely define the two most important roles: Central and Peripheral.

A Peripheral device periodically broadcasts advertisement packets to announce its presence and, if connectable, accepts an incoming connection request from a Central. Meanwhile, a Central device listens to advertisement packets and, when applicable, initiates a connection with a Peripheral. As an illustration, a smartphone can connect to a smartwatch to send notifications and collect sensor readings.

Note that, a Peripheral stops to advertise as soon as a Central is connected to it. Until the existing connection is ended, it thus undermines the possibility for other Centrals in range to discover and connect to it. Moreover, we emphasize that a BLE device cannot endorse both roles at the same time.

At the physical layer [38, Vol 6, Part B, sec. 2.1], BLE packets are split into four fields: the `Preamble`, the `Access Address`, the `Protocol Data Unit` (`PDU`) and the `Cyclic Redundancy Check` (`CRC`) (see Figure I.5). In the case of advertisement packets, the `Access Address` shall be set to the predefined value `0x8e89bed6` [38, Vol 6, Part B, sec. 2.1.2]. Divided into two parts, the `PDU` embeds a 2-byte header and a variable size payload.

On the one hand, the header describes the type of the `PDU` and indicates the class of the BLE device address (i.e. Public or random) [38, Vol 1, Part A, sec. 3.2.2]. On the other hand, the payload part contains the BD_ADDR followed by the advertising data. Finally, used for internal protocol management, the `Preamble` and `CRC` fields are automatically computed then filled by the Bluetooth stack prior to emit advertisement packets.
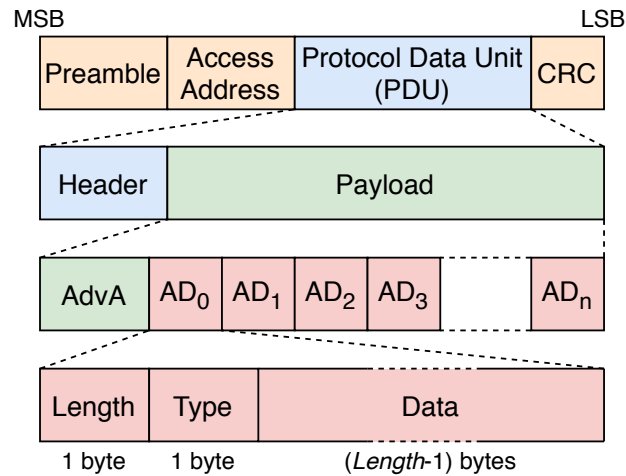
Figure I.5 – Format of the BLE advertisement packet. The payload part of the `Protocol Data Unit` (`PDU`) contains the `Advertising Address` (`AdvA`) field followed by Advertising Data (AD) structures. The `Length` field represents the length of the AD structure (excluding itself); `Type` specifies the nature of the following data and `Data` are the advertising data.

I.2.2.2   Device addressing and privacy features

During communications, a BLE device is identified by its BD_ADDR contained within the `Advertising Address` (`AdvA`) field of its advertising payload (see Figure I.5).

As part of the LE Privacy features [38, Vol 1, Part A, sec. 5.4.5], BLE has introduced random addresses allowing devices to replace their real BD_ADDR with temporary pseudonyms when broadcasting advertisement packets [40].

In addition to the globally unique Media Access Control (MAC) address, BLE supports three types of random addresses which can be identified leveraging the two MSB of the BD_ADDR [38, Vol 6, Part B, sec. 1.3.2]. As a consequence, there are four types of BD_ADDR in BLE:

— **Public**: the address uniquely allocated to the device by the manufacturer in accordance to the Institute of Electrical and Electronics Engineers (IEEE) specifications of MAC addresses [41, sec. 8.2];
— **Random Static** (MSB=`0b11`): a randomly generated device address that can be renewed after each power cycle, and that shall not change during the use of the device;
— **Random Non-resolvable** (MSB=`0b00`): a randomly generated address that can be renewed at any time, and that shall not be equal to either the Random Static nor the Public device address;
— **Random Resolvable** (MSB=`0b01`): an address composed of a 22-bit random number `prand` and a 24-bit `hash` produced by the hashing of `prand` with a 128-bit secret Identity Resolution Key (IRK). Shared between two BLE devices at the time

of pairing, the IRK allows the Central device to resolve the Peripheral address that would look random otherwise. For further details on the generation and resolution of such Random Resolvable addresses, we refer the interested readers to the Bluetooth Core Specification [38, Vol 6, Part B, sec. 1.3.2.2/3].

Based on their temporal persistence, we also classified those device address types into two categories:

— **Stable**: device addresses that are used by a device indefinitely or for an extended period of time (i.e. Public and Random Static addresses);
— **Private**: device addresses that are supposed to change frequently [1] (i.e. Random Non-resolvable and Random Resolvable addresses).

I.2.2.3   Advertisement

Aforementioned in Section I.2.2.1, *advertisement* is the name of the BLE discovery mechanism that allows Central devices to discover Peripherals in range. More precisely, it is used to broadcast connectionless data for applications as well as a prerequisite for a Central to set up a connection with a Peripheral.

I.2.2.3.1   Discovery mechanism

When a Peripheral is in advertising mode, advertisement packets are periodically broadcasted on each advertising channels. In addition to the frame header, advertisement packets can contain up to 31 bytes of data called *advertising data*. Those packets being broadcasted in clear, we draw the attention to the fact that their content can be thus collected and processed by any nearby device.

Furthermore, a Central can query a Peripheral by sending a directed scan request that triggers back a scan response. Transmitted over the three identical physical advertising channels, scan responses follow the same format as advertisement packets but carry different elements of information.

I.2.2.3.2   Advertising data

The BLE advertising payload consists of the `AdvA` field (equal to the BD_ADDR of the advertising Peripheral) followed by a sequence of one or more *Advertising Data* (AD) structures including items of information.

In particular, an AD structure is composed of a 1-byte field indicating the length of the

---

1. The Bluetooth Core Specification [38, Vol 3, Part C, App. A] recommends to renew Random Non-resolvable and Random Resolvable addresses at most every 15 minutes.

Table I.1 – Description of the most commonly advertised AD types.

AD type	Name	Description
0x01	`Flags`	Indicates discoverable modes and device capabilities
0x03	`Complete List of 16-bit Service Class UUIDs`	Lists 16-bit UUIDs of the supported services
0x09	`Complete Local Name`	Contains a string providing a user-friendly description of the device
0x0a	`Tx Power Level`	Indicates the transmitted power level of the advertisement packet
0x0d	`Class of Device`	Indicates the type of device (computer, phone, wearable, etc.)
0x12	`Slave Connection Interval Range`	Contains the preferred connection interval range of the Peripheral
0x16	`Service Data-16-bit UUID`	Consists of a 16-bit service UUID along with the data associated with this service
0x19	`Appearance`	Describes the physical representation of the device (sports watch, insulin pen, etc.)
0x26	`Transport Discovery Data`	Determines the organization and transport of the supported service
0xff	`Manufacturer Specific Data`	Contains custom data for manufacturer-defined applications

AD structure (excluding itself), followed by a 1-byte field specifying the type of the AD and finally, a sequence of up to 29 bytes of data (see Figure I.5).

Generally, AD structures embed device characteristics, capabilities and supported services, but also data coming from custom applications. Leveraging online resources of the Bluetooth Special Interest Group (SIG) [42], we noticed a total of 44 AD types, and listed some of the most commonly advertised in Table I.1.

With regard to the advertising payload, it can be either 1) set by the application or 2) automatically filled by the Bluetooth stack based on services configured into the Generic Attribute (GATT) profile of the Peripheral (see Section I.2.2.4). Between those two options, the single difference is that the first option allows the appliance to have full control over the advertising payload. Especially, it is useful to send proprietary data such as those that can be carried by the `Manufacturer Specific Data` AD structure (see Table I.1).

I.2.2.3.3 Service Universally Unique Identifiers (UUIDs)

Complying with RFC 4122 [43], a UUID is a 128-bit identifier that is mostly used to identify devices and resources. In the context of Bluetooth/BLE, UUIDs are leveraged to identify services [38, Vol 3, Part B, sec. 2.5].

Inventoried by the Bluetooth SIG [42], a Peripheral can rely on 12 AD types dedicated to the broadcasting of UUIDs to advertise its services. Among them, the most commonly practiced are the `Complete List of 16-bit Service Class UUIDs` and the `Service Data-16-bit UUID` AD types (see Table I.1).

For the sake of space saving into the 31-byte long advertising payload, some of those AD structures include the full UUID (i.e. 128-bit long), while others embed a truncated version (i.e. 16 or 32-bit long). To make this possible, the Bluetooth SIG adopted `XXXXXXXX-0000-10008000-00805f9b34fb` [38, Vol 3, Part B, sec. 2.5.1] as the standard base UUID where the 32 MSB (represented by `X`) are up to manufacturers.

Overlaying a 16-bit UUID in the `X` digits above, all Bluetooth SIG specified services [44] such as the `Heart Rate Service` and `Continuous Glucose Monitoring` then use this base UUID. As an example, the `Heart Rate Service` is advertised as `0x180d` (i.e. a 16-bit UUID) but actually represents `0000180d-0000-1000-8000-00805f9b34fb`, the corresponding 128-bit UUID.

Also, 128-bit UUIDs can be fully customized by manufacturers (see Table C.3 in Appendix C). Nevertheless, we point out that the generated UUIDs are for practical purposes unique. Furthermore, unlike most numbering schemes, their uniqueness does not depend on a central registration authority. As a consequence, RFC 4122 compliant UUID generators such as [45, 46] shall be then used to avoid collisions.

I.2.2.4   Generic Attribute (GATT)

In BLE, the Attribute (ATT) is a Client-Server stateless protocol based on *attributes* where devices can endorse each role regardless of their BLE role (i.e. Peripheral or Central) [38, Vol 3, Part F, sec. 2].

Data exposed by a Server are presented in a GATT profile which is a hierarchical structure of attributes allowing the transfer of information between a Client and a Server. Likewise, it provides a reference framework for all GATT based profiles covering precise use cases and ensuring interoperability between devices from different vendors.

As data related to applications and users must be formatted then sent according to its rules, this makes GATT a key element of the Bluetooth Core Specification. Note that, a GATT profile is only accessible after a connection has been established between two BLE devices meaning that they have already gone through the advertising process.

Identified by a UUID (see Section I.2.2.3.3), attributes within a GATT profile can be either *services* or *characteristics*. In the hierarchical structure, conceptually related *characteristics* are grouped below a same *service* (see Figure I.6).

In addition to their UUIDs, *characteristics* are composed of an attribute handle, a set
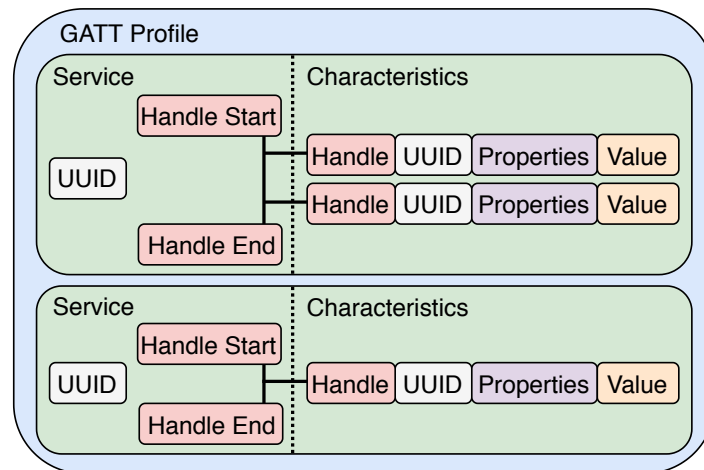
Figure I.6 – Structure of a GATT profile. Services are composed of a UUID along with two handles (`Handle Start` and `Handle End`) delimiting the hierarchically dependent characteristics. Characteristics are each constituted of a handle, a UUID, a set of properties and a value containing data.

of properties and a value. The handle specifies the position of the characteristic in the profile, while the value holds the actual data of the characteristic. Properties are metadata that specify which ATT operations (i.e. read, write, etc.) can be executed on each particular attribute and with which security requirements (i.e. encryption, authentication). Independently applied on each characteristic, such access permissions are mostly implementation specific[1].

Similarly, a *service* is identified by its UUID and associated with two handles (*Handle Start* and *Handle End*) specifying a range of *characteristics* that are hierarchically dependent from this service.

In fact, vendors are free to define their own services and characteristics, but the Bluetooth SIG has already set up a number of them [47]. For instance, the `Device Information` [48] service contains the `Model Number String`, `Software Revision String` and `System ID` characteristics. As another example, the `Battery Service` [49] embeds the `Battery Level` characteristic.

Lastly, the BLE includes security mechanisms such as encryption and authentication that can be used within the ATT protocol. Certain values of characteristics can be then exclusively accessed by an authenticated Client [38, Vol 3, Part C, sec. 10.3]. Note that, the value is the sole element protected by this feature. Therefore, the list of services and characteristics as well as the associated metadata (i.e. handles and properties) do not require any authentication to be accessed [38, Vol 3, Part G, sec. 8.1].

---

1. Properties are defined at the *Applications* layer of the Bluetooth stack (see Figure I.1).

I.3 Document structure

In addition to this introductory chapter that provides the prerequisites for understanding our studies, this manuscript is composed of five chapters.

To start, a state of the art dealing with radio signals, connected objects and their privacy aspects is established in Chapter II. Thereafter, Chapter III describes how *passive* and *active* eavesdroppers can leverage BLE advertising data and metadata to *physically track* devices and their corresponding owners over time. As the second investigated privacy threat, the exploitation of the Bluetooth/BLE wireless communications in order to *infer information* on remote users is presented in Chapter IV. Applied to the specific case of *Apple* BLE Continuity protocols, Chapter V discloses a collection of personal data leaks ranging from device technical characteristics to e-mail addresses and phone numbers of consumers. To finish, the limitations of our research along with its perspectives for future work as well as its industrial, social and media impacts conclude this document in Chapter VI.

Note that, throughout our work, we aim to illustrate that the wireless communications of the IoT can raise privacy concerns. In this direction, we set up and experiment attacks based on the Bluetooth/BLE radio communication standards. However, we highlight that the scope of this thesis can be further extended to wireless standards in general as they could imply similar privacy issues (see Section VI.2).

I.4 Contributions notice

Within this manuscript, each chapter results from scientific papers that are all co-authored with Mathieu Cunche [50], one of the supervisors of this thesis.

To define methodologies and protocols for experiments, Mathieu guided this research considering insights that I, Guillaume Celosia [51], provided. The design and prototyping of tools, the building of datasets through collections of radio signals, the processing of data as well as the software production only involved me.

Finally, the writing of the papers was fairly shared between us. To be transparent, this means that some reformulated parts of this document originally come from Mathieu.

Chapter II

State of the Art

This chapter reviews the state of the art relevant to the domains and techniques approached in this thesis. In particular, service discovery mechanisms used in wireless technologies are first introduced in Section II.1. Then, Section II.2 demonstrates that radio signals broadcasted by such discovery mechanisms are a source of privacy leaks. Physical tracking and device fingerprinting techniques along with their applications are detailed in Section II.3. Section II.4 discusses additional privacy threats that aim to infer users information based on their carried connected devices. Technical as well as legal existing privacy protections in wireless technologies are analyzed in Section II.5. Section II.6 presents the paradigm of the Internet of Things (IoT) before providing an overview of its deployment and privacy implications. To finish, the motivation for this thesis and its contributions to the state of the art are outlined in Section II.7.

II.1   Service discovery mechanisms in the wireless era

Service discovery is defined as the action of finding a service provider for a requested service. As a consequence, service discovery mechanisms are network protocols allowing the detection of services offered by devices on a computer network. As such, a network service can be any software or hardware entity that a user might be interested to use. For instance, printing, sharing photos and file transfer are network services.

In the context of wireless technologies, discovery of service [1] is provided prior to establish a connection between devices. Service discovery protocols are then specifically designed as the set of available services dynamically changes based on the proximity of mobile devices. For instance, the pairing of Bluetooth earphones requires the remote smartphone to determine 1) which Bluetooth profiles such as hands free and advanced audio distribution profiles are supported by the earphones, and 2) what protocol multiplexer settings are

---

1. The more general service set includes the device/network subset.

needed to connect to them. Service discovery protocols are thus a key component of wireless network systems. In particular, the discovery of services is an essential feature for the usability of ad hoc networks, and will ensure availability of services to users and applications. Since the massive development of ubiquitous computing, the service discovery paradigm arises bringing wireless communications as one of the important mediums of transmission of information or data to other devices.

In order to discover their surroundings, wireless devices can use either an active or a passive service discovery mode.

In the active mode, devices advertise their presence through the broadcast of radio signals to be connected. For instance, Bluetooth enabled devices emit *advertisement packets* crowded with data such as names, appearances and manufacturer specific data.

In the passive mode, devices passively listen to broadcasted signals announcing characteristics and capabilities of the corresponding peripheral. Back to the previous example, devices passively listen to advertisement packets broadcasted by nearby devices in order to connect to it.

Note that, as mobile devices are battery-powered, they prefer the active service discovery mode because of its reduced energy consumption with regard to the passive one. Moreover, it is the only mode that allows to reach hidden devices (i.e. devices that do not advertise their presence broadcasting radio signals) and speed up reassociation with them.

As a result, service discovery mechanisms are the first step toward wireless communications between devices. Here, *wireless communications* refer to the transmission of information over a distance without requiring wires or any other electrical conductors. As a consequence, information and data are transmitted over the air through electromagnetic waves such as infrared and, most of the time, radio frequencies. Intended distances can vary from centimeters for Near Field Communication (NFC) to thousands of kilometers for satellite communications, and thus encompasses various types of applications including smartphones, global positioning system (GPS), headphones, garage door openers, wireless keyboards, etc.

Figure II.1 presents a scale based classification of data networks from the Personal Area Network (PAN) to the Wide Area Network (WAN), along with corresponding ranges and wireless standards. In this thesis, the Wireless Personal Area Network (WPAN) will be only addressed through the Bluetooth technology.

WPAN, works like a PAN except that it uses a wireless communication medium instead of a wired one. Basically, it interconnects hand-held and mobile devices such as smartphones, headsets and smartwatches but also smarthome appliances [52]. In a more general way, it comprises all devices that are centered around a personal workspace.
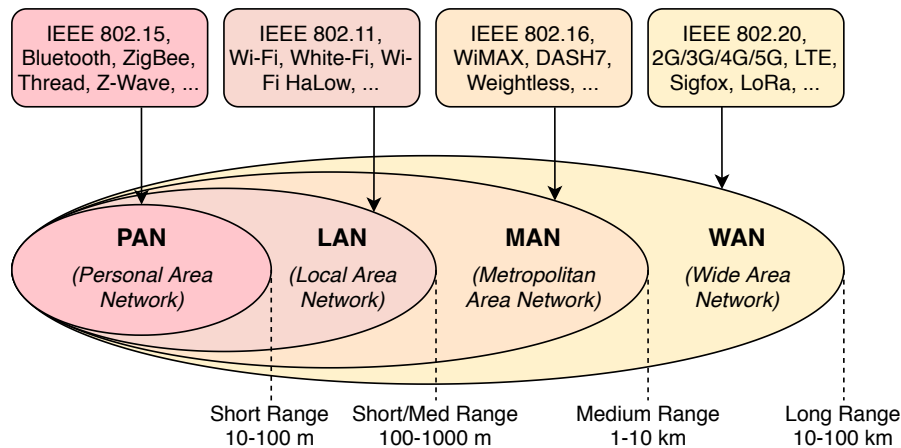
Figure II.1 – Scale based classification of data networks.

With regard to the range, it mainly depends on the device antenna capabilities but it does not usually exceed a hundred meters (see Figure II.1). As a result, Bluetooth, ZigBee, infrared or any similar wireless technologies can be part of a WPAN.

In the following, we will only deal with wireless communications relying on radio signals. As a consequence, we discard infrared and less common methods of communication using other electromagnetic wireless technologies such as light, magnetic fields and ultrasound.

II.2   Radio signals as a source of privacy leaks

Connectivity is a key element of connected objects that often rely on wireless communications through radio technologies. Indeed, compared to wired technologies, wireless technologies require a lightweight infrastructure and involve lower deployment costs.

Nowadays, connected objects support a variety of application domains bringing a diversity of constraints with regard to communication aspects. Range, bandwidth and energy consumption depict the three most important communication constraints that are satisfied leveraging radio technologies. Each of those technologies is characterized by different frequency bands, modulation, coding and protocol. Figure II.2 presents the most common radio technologies used by wireless devices along with their corresponding frequency bands and applications.

Some of the radio technologies are legacy technologies that have been used in other contexts (e.g. Bluetooth and Wi-Fi), while others have been specifically developed to match the need of connected objects (e.g. LPWAN and ZigBee). Although many of those technologies are based on standards with open specifications, there is also a number of proprietary technologies for which the full specifications are not publicly available. In particular, this is the case of the Low Power Device (LPD433) operating on the 433 MHz band and gathering
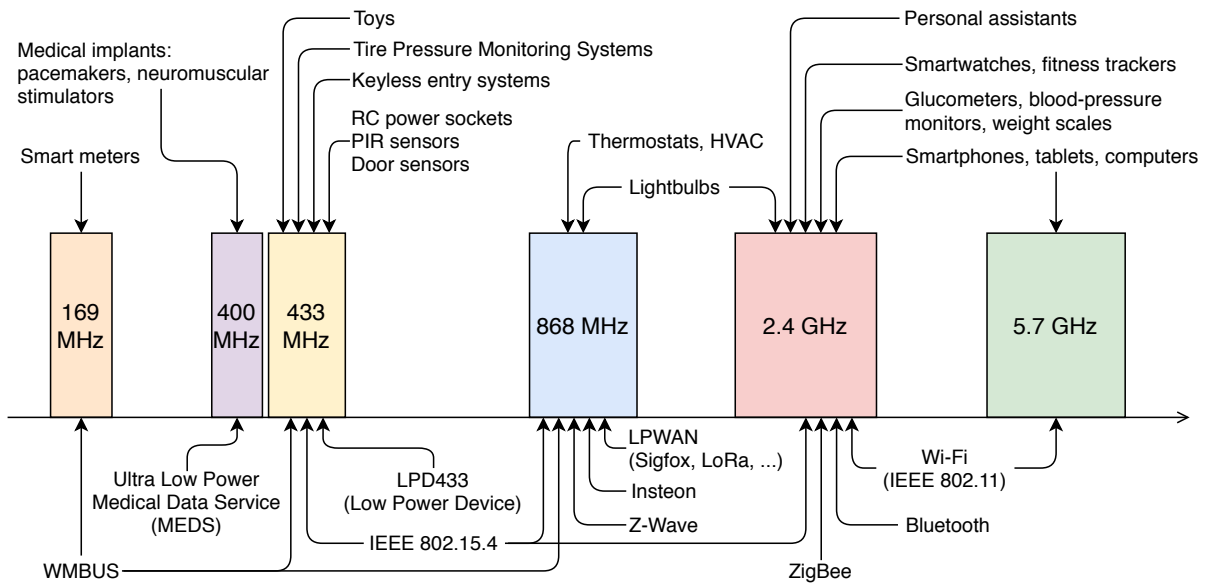
Figure II.2 – Example of radio technologies used by wireless devices along with associated frequency bands and applications.

devices such as door openers and keyless entry systems (see Figure II.2).

With the proliferation of wireless communicating devices, the amount of radio signals broadcasted by connected objects is bound to increase, and thus to be available over the air. Actually, the democratization of radio software [53, 54] and hacking tools [55, 56] facilitates the access to such signals. From now, anyone with basic computer skills can then exploit radio communications and trigger attacks.

As a result, most wireless technologies have addressed this threat through protection mechanisms such as encryption that guarantee the confidentiality of data transmitted over the network. However, even if the content of wireless communications can be protected, metadata (i.e. source/destination addresses, size and frequency of messages) are available in clear and can leak personal information. In fact, the existence as well as characteristics of radio communications can be collected and analyzed to infer personally identifiable information (PII).

For instance, radio signals can be used to uniquely identify a device, either because they contain a unique identifier or hold enough information to create a fingerprint of this device. Based on this information, it is thus possible to track the whereabouts of the person carrying the wireless communicating device. *Physical tracking* is the name of this privacy threat that commercial applications already exploit to track individuals leveraging Bluetooth and Wi-Fi device addresses [57, 58].

Beyond physical tracking, radio signals often contain enough information to identify the nature of the emitting device. Indeed, they generally include elements such as device names, appearances and service related data revealing the type of device or its purpose [59].

Furthermore, features of the radio traffic can be characteristic of a given application [60]. For instance, a surveillance camera will generate a continuous and high rate transmission while a smart outlet will broadcast signals based on sporadic actions of the user [61].

Moreover, because it is dedicated to a specific application, the radio technology itself can reveal the nature of the communicating device. For instance, smart meters and therapeutic medical implants are respectively using the Wireless Meter Bus (WMBUS) on the 169 MHz band and the Medical Data Service (MEDS) on the 400 MHz band (see Figure II.2). Therefore, by monitoring and analyzing features of radio signals, it is possible to build an inventory of the connected objects installed in a household or carried by a person.

Based on such inventory, it can be then possible to derive a detailed profile of the users. For instance, medical and health monitoring appliances can betray a specific medical condition: a blood-pressure monitoring system can imply a cardiovascular disease while a glucometer is usually associated with diabetes. The wealth of a household can also be inferred based on the price range of the detected devices. As an example, Bertrand and Kamenica demonstrated [62] that the ownership of *Apple* devices is a reliable indicator of wealth.

Finally, the observation of radio communications in a smart environment can be used to infer the activity of its users [63]. For instance, connected appliances in a smarthome such as light bulbs, air conditioning systems and voice assistants will only broadcast signals when they react to user presence or when they are in use. As a result, the emission or lack of those signals can reveal the presence of people in the house. Furthermore, if those signals can be associated with specific appliances, it can be possible to draw a detailed picture of the activity in the house: light turned on, fridge opened, kettle started, garage door opened, etc.

To summarize, even though data carried over the radio link can be protected by security mechanisms such as the encryption, characteristics and metadata associated with radio signals can be rich sources of information to infer private data.

II.3   Physical tracking and device fingerprinting

The fact that most connected devices may betray their presence through the broadcasting of radio signals can be exploited by eavesdroppers to capture and analyze mobility traces of corresponding users in the physical world. As such, the process of device fingerprinting is part of physical tracking as it intends to gather information about a targeted device for the identification purpose. In the following, we focus on those complementary privacy threats that both aim to track a person during its journey.

Physical tracking is defined as the activity that consists to follow the whereabouts of an individual along time in the physical world. In order to better understand the purpose

of radio based physical tracking systems, an overview of radio geolocation is first provided.

Geolocation consists in positioning an object in space by measuring its distances from reference points whose positions are known beforehand. As an example, the distances from three distinct points are sufficient to determine a position on a plane. Another famous example is the Global Positioning System (GPS): reference points are satellites whose positions are computed in real-time, and the distances between the object and those satellites are then deduced from the travel times of radio signals broadcasted by this object.

Based on the GPS approach, it is then possible to divert radio technologies from their main uses for geolocation purposes. For instance, even if the Bluetooth was not developed for localization at first, some solutions relying on this technology have been proposed for indoor asset tracking [64]. By using scattered Bluetooth enabled sensors as reference points, the Received Signal Strength Indication (RSSI) is then used to evaluate the distance to the signaling transmitter from the received signal strength. Also known as *trilateration*, this principle of position determination allows to determine the position of a Bluetooth terminal over an area of interest.

As a result, radio geolocation can be done in two ways. The first case corresponds to the previously described system where radio enabled devices leverage the deployed wireless infrastructure to determine their position without using the GPS module. In particular, services provided by the World Wide Web Consortium (W3C) Geolocation Application Programming Interfaces (API) standardize an interface to retrieve the geographical location information from a device. To this end, they use a request system along with a database containing positions of Bluetooth monitoring sensors on a global scale. Note that, this type of service is also provided by companies such as *Apple*, *Google*, *Skyhook* and *Mozilla* through its Mozilla Location Service (MLS) [65]. Likewise, to return a more accurate location information, other sources of information such as radio-frequency identification (RFID) and Global System for Mobile communications (GSM) cell identifiers can be correlated.

The second case of radio geolocation corresponds to a system that passively listens to wireless communications in order to detect and track mobile devices having their wireless interfaces enabled. In addition to the RSSI, each radio enabled sensor collects incoming wireless signals to extract a unique identifier such as a Bluetooth device address along with other technical information. Collecting sensors then report those extracted data to a remote server where they can be stored for further analysis such as aggregating statistics and estimating mobility of individuals.

Figure II.3 presents a Bluetooth based physical tracking system where RSSI and identifiers embedded in wireless signals broadcasted by nearby devices are leveraged by several sensors to accurately [1] determine the position of the involved device. With regard to the first case, it is important to mention that this type of physical tracking system does not need

---

1. The accuracy of such a system is claimed by *Apple* to go down to a meter [66].
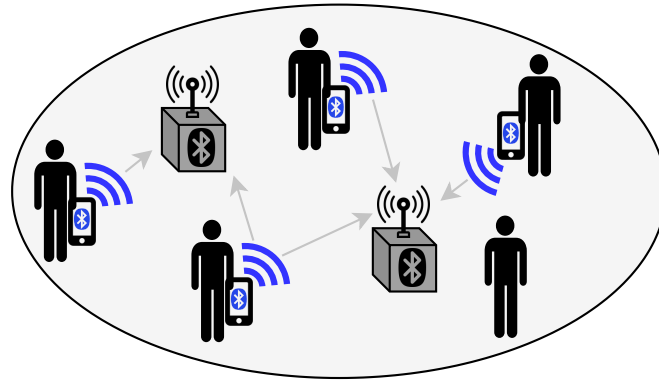
Figure II.3 – Representation of a Bluetooth based physical tracking system.

the collaboration of remote devices. Especially, this means that it is passive and does not require to install an application on wireless devices in order to be detectable by the system.

Therefore, we are currently witnessing the emergence of radio based tracking systems that record large-scale movements of individuals thanks to signals broadcasted by their carried devices [67]. Leveraging the deployed ubiquitous infrastructure [1], it is then possible to exploit wireless communications for analytics and profiling purposes in monitored areas. In this spirit, Nguyen et al. demonstrated [68] that video and radio technologies can be jointly used to track and monitor human activities. Moreover, physical tracking have diverse applications such as digital surveillance [69] and road monitoring [70] that are mostly performed without consents of targeted persons. As such, it might violate the privacy of the individuals.

Device fingerprinting is a process used to identify a device based on artifacts stemming from its specific software and hardware features. Also called *signature*, a device fingerprint is thus a representation of a stable pseudo-identifier built from collected information of such artifacts.

Device fingerprinting has a number of useful applications. For instance, it can control digital rights management (DRM) leveraging serial numbers assigned to the device hardware [71], prevent bank fraud by identifying whether an online banking session has been hijacked [72] and detect network intrusions [73]. In the following, we focus on the tracking applications relying on wireless device fingerprinting.

Tracking and analytics are two of the many possible goals of wireless device fingerprinting. In particular, in the retail sector, the visitor analytics has become a key concept to provide businesses with greater visibility into visiting habits of their stores. Those profiling solutions can be also leveraged to identify and track customers across shelves gauging their visit duration as well as their willingness to buy certain products rather than others. This

---

1. As an illustration, some Wi-Fi access points can be turned into Wi-Fi sensors.

information can be then exploited to reorganize shops toward influencing customer sales.

As for service discovery mechanisms (see Section II.1), device fingerprinting techniques can be either active or passive. Passive fingerprinting requires the eavesdropper to observe the wireless traffic generated by its targeted device, while active fingerprinting techniques require him to interact with the remote device (i.e. by initiating a connection or, at least, sending packets to the target).

Whether active or passive, the idea behind wireless device fingerprinting techniques is to extract specific device information such as Bluetooth advertised elements or Wi-Fi service set identifiers (SSIDs). The extracted information are further processed to compute a unique identifier of the targeted device thus forming the fingerprint of the device. Note that, information constituting the fingerprint can be extracted from both the software and hardware features.

In fact, even if same types of device (e.g. the same brand of smartphone or the same model of smartwatch) are carried by many people, each of them uses different configurations.

At the software level, two smartphones of the same model running the same version of their operating system (OS) can have different radio identities based on the customization of their features. This can then lead to the broadcast of wireless signals that are crowded with different data. For instance, this is the case of Bluetooth signals broadcasted by *Bose* headsets where device names can be customized to include the names of their corresponding owners (see Section IV.3.2.3).

From the hardware side, even though those two previous smartphones are of the same model, they can embed different wireless chipsets. As a consequence, radio related characteristics such as the channel selection algorithm and the transmission power of emitted signals can differ. This process of device fingerprinting at the radio hardware level is also known as *radio frequency fingerprinting*, and is commonly used by cellular operators to prevent cloning of cell phones: a device having the same software identity as the original one is declared as being cloned if its radio fingerprint differs. Furthermore, we point out that it is sometimes possible to distinguish two devices using the same chipset model because of the imperfections induced in the manufacturing process [74, 75].

Finally, the only requirement to uniquely distinguish devices over time is to compute fingerprints that are both sufficiently stable and diverse. In practice, the improvement of the stability of the fingerprint tends to impact its diversity and vice versa [76]. As an example, the customization of Information Elements (IEs) carried by Wi-Fi probe requests increases diversity [77]. However, at the same time, this reduces stability as the whole device fingerprint changes when the collection of advertised IEs is modified. In other words, this means that neither stability nor diversity is fully achievable.

II.4   Inferring users information in the physical world

The previous Section II.3 described *physical tracking* and *device fingerprinting* as first types of attack that aim to track individuals based on radio signals broadcasted by their carried devices. In this section, we highlight two other types of radio based attack jeopardizing the privacy of users as well: *activity inference* and *inventory attacks*.

Activity inference aims to recognize the activity of an individual through series of observations of radio signals emitted by its wireless communicating devices.

Due to its strength in providing personalized support for many practical applications including quantified self and energy conservation in smarthomes, activity inference is one of the new privacy threats that was introduced with the massive deployment of connected objects.

The effectiveness of such an attack mainly relies on the diversity of radio signals that exist. For instance, a remote control for a smart television (TV) will broadcast a different signal than a remote control for a connected speaker. As a result, by observing characteristics such as the frequency band and the modulation of those radio signals, the attacker can then distinguish when his target is watching TV from when he is listening to music.

Moreover, the correlation between temporal patterns and type of collected radio signals allows to profile a targeted individual revealing its habits as well as its traits of character [78]. To illustrate this statement, a person who uses its connected vacuum cleaner every day is probably maniac as he likes when things are always cleaned and tidied. An other person who will drink coffee ten times a day is certainly more stressed than when only two coffees are brewed from its wireless coffee machine, for instance.

In any case, activity inference provides a lot of information to eavesdroppers that can be further exploited for other targeted attacks such as phishing, door-to-door selling and even thefts. Indeed, the absence of broadcasted radio signals can be also leveraged to betray a vacant smarthome.

Inventory attacks consist in establishing a list of connected objects associated with an individual in order to infer personal characteristics.

Note that, the inventory attacks and activity inference are strongly related as knowing the type of device that emitted the radio signals can be the prerequisite of the activity inference. As a consequence, the inventory attacks can be assimilated as the first step toward establishing a complete profile of an individual.

With the proliferation of application domains for connected objects, users have surrounded themselves with devices corresponding to an increasing number of use cases (see the list of Bluetooth device classes [79], as an illustration). Therefore, we are witnessing the emergence of inventory attacks enabled by wireless communications. Actually, for the needs

Table II.1 – Non-exhaustive list of personal information inferred from types of connected devices.

Type of device	Inferred personal information
Hearing aids	User has hearing deficiency
Glucose monitor	User has diabetes
Weight scale	User controls his weight
Blood pressure sensor	User has heart disease
Fitness tracker	User is a sportsman
Cycling speed sensor	User is a cyclist
Pets tracker	User owns a pet
Toy for children	User has children
Coffee machine	User drinks coffee
High quality speaker	User loves music

of service discovery mechanisms (see Section II.1), wireless communicating devices mostly emit identifiers in clear revealing their natures. For instance, Bluetooth smartwatches manufactured by *Samsung* and *Asus* respectively broadcast `GALAXY Gear` and `VivoWatch` as an explicit identifier (see Table C.1 in Appendix C).

The nature of the owned devices can then serve as a basis to infer personal information. As an example, the presence of hearing aids can reveal that the corresponding owner has hearing deficiency, while the usage of a cycling speed sensor indicates that he is a cyclist. Table II.1 presents a non-exhaustive list of possible personal information inferences leveraging types of connected devices.

Once the inventory of devices is established, the next step of the attack is to draw a profile of the targeted user. Some devices such as fitness trackers and speakers can be associated with special uses that can reveal points of interests (e.g. in those cases, sport and music). More worrisome, presence of devices such as Bluetooth insulin pens and glucose monitors can leak a medical condition of the owner (e.g. in this case, diabetes).

Last but not least, a standard of living can be also derived from the information provided by the inventory of devices. As an example, the wealth of a household crowded with connected devices of famous brand such as *Apple* or *Google* is probably higher than the one with a cheap smart outlet alone.

To finish, similarly to physical tracking applications, the inference of personal data through inventory attacks can be exploited by advertisers and marketers to trigger targeted advertising campaigns. As an example, an offer can be specifically designed for a set of users in order to influence them to supplement their collections of devices. In this spirit, the advertising campaigns can be adapted according to the brands of the devices owned by the targeted persons. For instance, advertisers will more likely target *Apple* fans to announce the release of a new *Apple* iPhone smartphone as they will be more receptive to it than *Huawei* lovers.

II.5   Privacy protections in wireless technologies

Radio signals have been difficult to reach as the required hardware is expensive and mainly intended for professionals. As a consequence, the absence of radio enabled equipment on the consumer market makes vendors obtain a pseudo-security of the data exchanged over the air. However, it is important to remind that this type of pseudo-security relying on technical lacks is referenced as one of the worst observable practices in computer security.

Allowing to understand most mechanisms of wireless communications, the evolution of technologies has provided the population with modest equipment. Geneiatakis et al. have even shown [80] that spying has become a new game in which anyone, with basic computer skills, can take part.

Therefore, such a democratization of software and radio hacking tools made manufacturers to take into account privacy protections within the development of their connected objects. Moreover, the enforcement of the General Data Protection Regulation (GDPR) puts both legal and economic pressures on vendors regarding the privacy preservation of customers (see Section II.6.4).

Aiming to give control to individuals over their personal data, the GDPR is challenging as it provides a vague definition of *personal data*. For instance, even though the process of device fingerprinting only collects technical data on wireless devices (i.e. not PII such as e-mail addresses or names of users), the resulting fingerprints fall into the category of personal data. Actually, those bits of information relate to individuals and can be further exploited to identify them. In the context of the GDPR, a personal data is thus defined as any information that can be leveraged to identify an individual, wireless device identifiers being part of it.

Based on the general guidelines about consent of users and legitimate interests provided by the GDPR, several efforts on wireless communications have been conducted to protect users against privacy breaches.

To prevent third parties from leveraging device addresses in order to track users, the address randomization has been introduced [77, 81]. Behind this concept, the idea is to frequently replace the address of the device (i.e. the Media Access Control (MAC) address) with a new random and temporary one. As a result, a tracking system will no longer be able to link radio signals broadcasted by a single device and thus to estimate the mobility of individuals. In fact, while tracking was a known threat since years [82, 83], MAC address randomization has been long promoted before vendors started to consider it in commercial devices [84, 85]. Note that, address randomization has begun to be implemented in 2014 [86] as the main feature of various operating systems, namely *Google* Android since Android 6 [87], *Apple* iOS from iOS 8 [88], Linux drivers since kernel 3.18 [89], *Microsoft* Windows from Windows 10 [90], and has even made its way to wireless standards [91, 38].

Due to the lack of standardization, device fingerprinting is more difficult to tackle as it is mainly dependent from software and hardware features of devices. Besides, if software features can be addressed by manufacturers at the kernel or firmware level, the hardware remains impossible to modify as it includes intrinsic characteristics of wireless chipsets.

As opposed to the technical point of view, the only legal requirement is then to ensure that companies using device fingerprinting techniques are operating according to laws. This means that either providing a legitimate interest to collect and process fingerprints of nearby devices or asking consents of their corresponding owners is mandatory not to collide with regulations. In this case, the legitimate interest will be likely reserved for particular applications such as identity theft and fraud prevention, while consents of users have to be collected by companies that leverage device fingerprinting for advertising purposes, for instance.

To protect users against activity inference, some works [92, 93] based on radio technologies proposed countermeasures that consist to obscure the real network traffic with 1) the injection of a synthetic one and 2) the addition of random delays in the transmission of sensor signals. Nonetheless, as for device fingerprinting, those approaches need to be extended to deal with the case where wireless communications themselves are potential sources of privacy leaks.

Compared to the other threats, inventory attacks are relatively new as they were raised accordingly to the massive deployment of single-purpose wireless devices. For instance, cycling sensors and glucometers are respectively used by cyclists and diabetic individuals. As a consequence, there is little work that has been done to design countermeasures.

To put in a nutshell, thanks to technological advances, the landscape of wireless communications has become accessible by the general public. From now, characteristics of radio technologies can be leveraged to breach privacy of users, and raise several issues from physical tracking to inventory attacks. Moreover, even if privacy protections such as MAC address randomization have been widely adopted, they possess limitations [1] that could negate their purposes. Likewise, there is still a lack of countermeasures applied to privacy threats where wireless communications themselves are sources of privacy leaks. Powering a plethora of privacy invasive purposes such as targeted selling, those issues also face a number of challenges due to enforced privacy rules, like the GDPR.

---

1. Especially, the address randomization has been attacked a lot [94, 95, 77, 96, 81, 97, 1].

II.6 Overview of the Internet of Things deployment and its privacy implications

II.6.1 The Internet of Things paradigm

Back to 1999, Kevin Ashton invented the *Internet of Things* (IoT) during his work at Procter & Gamble to promote the RFID technology. With the devices/people ratio growing from 0.08 in 2003 to 1.84 in 2010, Cisco Systems then defined [98] the IoT as *"the point in time when more 'things or objects' were connected to the Internet than people"*. Therefore, Cisco estimated that the IoT was born between 2008 and 2009.

Nowadays, the definition of the IoT has evolved due to the convergence of technologies such as commodity sensors and real-time analytics [99]. In fact, all wireless sensor networks, embedded and automation/control systems contribute to enable the IoT. Moreover, in the consumer market, the IoT is, most of the time, assimilated with smarthome appliances such as lighting fixtures and thermostats that can be controlled via remote devices such as smartphones and tablets.

In this thesis, we study the IoT from the point of view of wireless networks. As a consequence, we define the IoT as a term used to designate all the physical devices, generally equipped with sensors and/or actuators, that are linked and communicate through wireless networks.

Over the past decade, the number of connected devices has grown quickly. As of today, the number of IoT devices is estimated to 31 billion units, and it is forecasted that this number will reach 75 billion in 2025 [23]. This booming market affects all domains from the agriculture to healthcare, through transportation and entertainment.

Among domains of application in which the IoT is widely adopted, there are several in which connected devices are interacting with people. For instance, thermostats and lightbulbs in a smarthome adapt the temperature and light to occupants, while smart meters collect and report the energy consumption. In healthcare, medical sensors and implants monitor health and provide medical assistance to patients. In fitness, wristbands record and report physical activities of sportsmen. In vehicles, sensors report pressure of tires, acceleration and speed to the control unit, while the later communicates with the infrastructure and other vehicles. Figure II.4 presents a typical example of connected devices that can be found in a household.

Thus, with the rapid adoption of the IoT, the number of applications involving consumers and the level of interaction with the user are both bound to increase. In fact, a growing portion of IoT devices are created for consumer use. Behaving as a team of assistants, consumers can then benefit from the information evaluation and data analysis of the IoT that enhance the way they live, work and play. However, the development of the IoT is not going without concerns.

Figure II.4 – Example of IoT devices surrounding users.

To be efficient, IoT devices need to store and process data to trigger actions and scenarios defined by users. For this purpose, connected objects collect data about activities of users before being sent to remote servers of IoT manufacturers in order to enable various services. This leaves the door wide open for privacy and security threats where the collection of the consumer data can be further exploited to infer personal information. As an example, Molina-Markham et al. demonstrated [100] that high resolution data reported by smart meters can be used to infer the activities of inhabitants in a household.

II.6.2   Radio communications in the Internet of Things

As described in Section II.6.1, behind the IoT paradigm are devices that help people to take decisions and automatize actions through information evaluation and data analysis. Nevertheless, to acquire data, one of the crucial components of the IoT is the communication network. On the one hand, the communication network of the domestic IoT can be seen as the gateway between an IoT device and a software platform such as *IF This Then That* (IFTTT), for instance. On the other hand, the industrial IoT devices mostly form a local network topology where they dynamically and non-hierarchically connect to directly exchange data between themselves.

The rapid development and evolution of wireless technologies have the potential to provide flexibility and mobility to the billions IoT devices. In particular, data speed rates, wide ranges and low power capabilities of wireless technologies such as Bluetooth, Wi-Fi and 4G enable most mobile IoT applications. Indeed, reinventing not only the way individuals connect to the Internet but also the way surrounding devices can be leveraged in everyday activities, the IoT devices are mainly carried by people over time. As a consequence, wireless is becoming the primary form of communication to face with the high demand for connectivity generated by IoT devices.
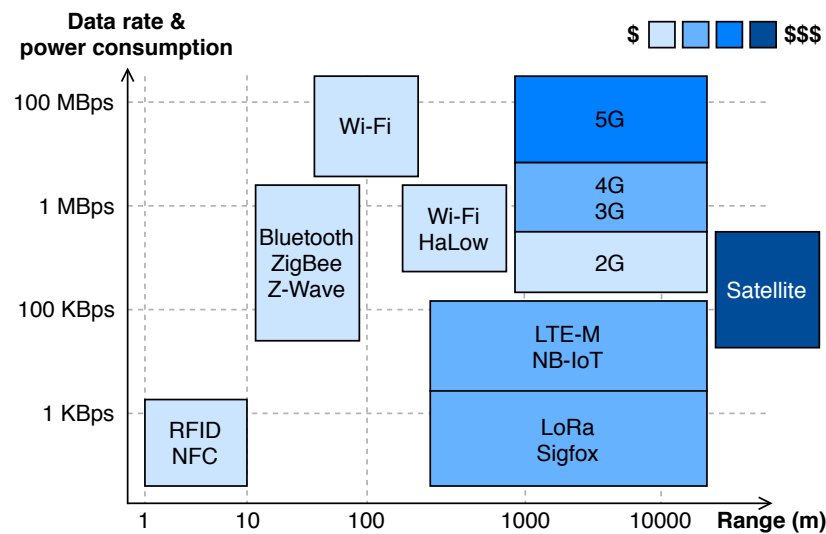
Figure II.5 – Representation of wireless technologies used by IoT devices.

By definition, wireless communications involve the transmission of information over a distance without using wires or any electrical conductors. Note that, the transmission of information can range from a few meters (e.g. an infrared remote control for a speaker) to thousands of kilometers (e.g. radio communications). To transmit and receive data, IoT devices can leverage various wireless technologies such as Wi-Fi, ZigBee, Z-Wave, 4G and Bluetooth, as presented in Figure II.5. All those technologies are used depending on the application offered by the device. For instance, ZigBee was specifically designed for Machine-to-Machine (M2M) networks (e.g. a thermostat sending the temperature to an application software that will adjust the heater in the room based on this information), while Bluetooth is more commonly used in consumer applications such as fitness trackers and smartwatches.

As wireless technologies enable IoT devices to transmit and receive data without a physical infrastructure, they have pros and cons. Undeniably, reducing the wire restriction is one of the benefits of wireless compared to wired devices. Inter alia, wireless technologies also enable accessibility where ground lines cannot be laid.

Other benefits include the dynamic network formation, easy deployment and low cost. Indeed, wireless networks do not require an elaborate physical infrastructure or maintenance practices, hence reducing the cost. In addition, using wireless technologies, people can exchange data from anywhere, at anytime, regardless of their locations. This means that there is no need to physically connect the device to be able to send and receive messages. Finally, the constant connectivity and data speed rate also ensure that people can quickly respond to emergency situations.

Despite those benefits, wireless communications raise several issues. One of the most common issues is interference. In fact, the speed and viability of the wireless signals drop as more and more devices are using the same frequency. This can lead a wireless network

to malfunction and, in extreme cases, to the complete loss of wireless communications. In addition, compared to a wired network, getting a consistent coverage and data speed rate can be difficult as wireless signals are attuned by obstacles such as walls and doors, for instance.

To finish, flexibility and mobility provided by wireless technologies raise issues related to security and privacy. Actually, contents of wireless communications are more exposed to attack than wired ones. As a result, if not properly secured, data and metadata broadcasted by IoT devices can be exploited to threaten security and privacy of users (see Section II.2).

II.6.3   Privacy aspects of the Internet of Things

Frequently associated with the collection of large amount of data, IoT devices become more prevalent raising significant challenges with regard to the privacy protection of users [101]. To support this, the European Union (EU) commission even highlighted [102] security and privacy as major IoT research challenges.

To make the matter worst, a Pew Research Report discussed [103] privacy issues of the IoT and states that such a massive deployment of those connected objects will result in a world with a *"small class of 'watchers' and a much larger class of the experimented upon, the watched"*. Into the same research report, Marc Rotenberg, the Electronic Privacy Information Center (EPIC) president, puts an emphasis on the fact that the IoT users *"are just another category of things"*.

Indeed, aiming to provide better services that fulfill personal preferences in order to improve the user experience, information collected by the IoT can be also leveraged by companies to draw detailed profiles of individuals.

In this spirit, many research have shown how data collected from the IoT can be used to infer personal information. One of the most famous examples is how readings from a smartmeter can reveal the activity of people leaving in a household [100]. In addition, with the emergence of wearable devices and advent of quantified self, users collect data such as calories burned, number of steps and sleep quality on their daily activities that can further threaten their privacy [104, 105].

Another related field of application is health, which is by nature sensitive and for which the use of connected devices can expose the users to privacy threats [105]. Moreover, Beresford and Stajano showed [106] that many sensitive information about a user such as its home location, religion and health issues can be inferred through the collection of location data, leveraging dedicated trackers or smartphones. Note that, in addition to localization purposes, we draw the attention to the fact that the geolocation data collection is sometimes necessary to enable the corresponding IoT applications (e.g. indoor navigation, fitness tracking, weather alerts, etc.).

As a consequence, data collected through IoT devices can be used to generate enriched information on their owners. Even if this problem of data confidentiality exists, we point out that we do not further consider it in our work. In other words, whether they are protected by a security mechanism or available in clear, data carried by radio signals are not of interest within our studies.

Nevertheless, we aim to demonstrate that, beyond data, the characteristics and metadata associated with the signals can be a rich source of information to infer personal data. To better illustrate this statement, a passive attacker leveraging radio signals broadcasted by smarthome devices can learn when occupants are waking up (when the lightbulb in the kitchen is turned on), when they brush their teeth (thanks to smart toothbrushes), when they eat (thanks to the smart fridge), when they receive visitors in their house (thanks to the smart doorbell), etc.

In this thesis, we address privacy challenges within the wireless communications of the IoT. To this purpose, we will then focus on the metadata associated with those communications, and not the actual data conveyed by the radio link. For instance, in the case of a connected glucometer sending a blood sugar level over the air, we will consider the metadata included in the wireless communications such as the type of the device (i.e. a glucometer), its identifier (i.e. a device address), as well as its time and duration of use, excluding the blood sugar level.

II.6.4   Privacy in a connected world

The ubiquitous nature of the IoT increases attempted attacks aiming to extract personal data. To guarantee both the liability management associated with collected data and the information privacy related to the use of IoT devices, legal policies and frameworks have been designed.

Indeed, privacy protections get a lot of attention because of EU regulations such as the GDPR and the ePrivacy Regulation (ePR). Note that, those regulations both aim at strengthening the data protection within the EU increasing, at the same time, the pressure on companies with regard to the protection of their customers.

First, the GDPR, which is setting a general framework for data protection, has become active since May 25th, 2018. From its provisions, four key principles emerged, namely *consent*, *transparency*, *personal rights* and *accountability*.

To address the *consent*, Article 7 stipulates that *"consent should be given by a clear positive act by which the data subject expresses in a free, specific, informed and unambiguous manner his consent to the processing of personal data concerning him, for example by means of a written statement, including electronically, or an oral statement"*. Regarding the *transparency*, Article 12 states that organizations must provide individuals with clear and unambiguous information about how their data are processed. Last but not least,

new *personal rights* have appeared within the GDPR such as the right to be forgotten for all users and the right to data portability allowing an individual to recover, in a reusable form, the information he has provided. Finally, the principle of *accountability* gathers all the measures making companies more responsible when they process personal data.

Actually, most companies are affected by the provisions of the GDPR which objective is to strengthen the supervision of practices related to the collection and use of personal data. Moreover, it is important to note that the GDPR only concerns the protection of personal data related to natural persons. This means that processing data related to legal persons is out of the scope, unless they are required to collect data on representatives of legal persons. For instance, data collected from business cards such as e-mail addresses and phone numbers fall into the scope of the GDPR, while data such as company names and corporate objects are excluded.

Second, focusing on the case of privacy in digital communications, the ePR [107] has been introduced as an extension to the GDPR. At the origin, the GDPR and ePR should have been enforced at the same time. However, the ePR has attracted criticisms by industry lobbying that postponed its enforcement to 2020.

Indeed, French press publishers and representatives of the telecoms and digital industries deplored the ePR and asked for its revision. Similarly, the American Chamber of Commerce to the EU considered that the ePR hinders innovation. *Amazon*, *Microsoft*, and the French branch of the Interactive Advertising Bureau (IAB) also shared this point of view. Finally, the Developers Alliance, including *Facebook*, *Google* and *Intel*, reported that the ePR could cost businesses in Europe more than 550 billion euros.

Compared to the GDPR, the ePR goes further with regard to the processing of personal data. In fact, if the GDPR focuses on personal data, the ePR protects all forms of communication. This means that the ePR does not only apply to personal data, but also to the content and metadata of communications such as network identifiers (e.g. device addresses) that can be further leveraged for physical tracking, targeted advertising and personalization of content. Also known as *opt-in*, the ePR thus generalizes the concept of explicit consent, and has same levels of sanction as the GDPR in the case of an infringement. In addition, the ePR also applies to legal persons, unlike the GDPR.

To tackle privacy issues raised by the IoT, regulation and protection authorities have started taking steps. In particular, the independent French administrative regulatory body Commission Nationale de l'Informatique et des Libertés (CNIL) has published [108] an extension of its Privacy Impact Assessment (PIA) framework dedicated to the IoT. This document explicitly mentions several risks associated with wireless communications in the IoT including *"remote detection of electromagnetic signals"*. Similarly, the opinion of the G29 (Working Party of Article 29) on the Recent Developments on the Internet of Things [109] noted the tracking risk associated with the IoT. Especially, it is mentioned that *"wearable things kept in close proximity of data subjects result in the availability of a range of other identifiers [...] allowing data subject location tracking"*.

In parallel, consumers become increasingly aware about the privacy implications of IoT devices. This can be characterized by the fact that a growing number of people are taking privacy into consideration within their choices of consumption. As a result, it becomes a common thing to adopt products including privacy preserving features such as *Qwant* [110], a search engine that does not collect information on its users. Furthermore, high-tech companies are integrating privacy protection features in their flagship products. As an example, the *Apple* iPhone smartphone includes a protection against the radio based physical tracking since the release of *Apple* iOS 8. To finish, a 2016 survey [111], describing the relationship French people have with connected objects, pointed out the fact that the confidentiality of personal data is one aspect that can curb the acquisition of a connected object.

From now, the IoT vendors are subject to both legal and economic pressures regarding privacy preservation of their consumers. Hence, they are bound to consider privacy protections when developing their products, in conformance with regulations such as the GDPR and ePR that advise the *privacy by design* approach. Note that, the concept of *privacy by design* requires to think about the protection of personal data before developing a product or a service.

II.6.5 Concluding remarks and open questions

IoT devices are everywhere, to the point of blending into daily lives [112]. In a world where activities and behaviors are increasingly monitored, it is important that manufacturers and implementers think about how to inform their consumers with regard to the collection and usage of their personal data. To this end, previous sections highlighted the needs for both regulations and technical countermeasures to bridge the gap between the data leaked by the IoT through radio communications and the personal rights of individuals.

Even if the legal side is not addressed by the studies carried out in this thesis, questions such as "who collects my personal information ?", "what are personal information collected ?", "how are they stored and used ?", "to whom are they disclosed ?" and "for what purposes ?" shall be clearly covered by privacy policies of companies, but also by regulations and protection authorities. In addition, transparency about data collection is as important for the trust in IoT vendors as for the relationships with governments. Indeed, even if commercial purposes are the first that come to mind, it is not without thought that the wealth of information on individuals collected through their carried IoT devices can attract the interests from governments [113, 114].

From the technological point of view, the massive deployment of the IoT devices has not been followed by significant changes in governance models for privacy. Moreover, even if MAC address randomization has been introduced as a countermeasure to address the radio based physical tracking, it can still exist misimplementations that defeat this anti-tracking feature. To date, little attention has been paid to technically tackle the previously discussed privacy threats such as device fingerprinting, activity inference and

inventory attacks. In fact, if each threat individually leaks a small amount of data, in aggregate they can be leveraged to draw complete profiles of users. This is more worrisome when we have in mind that those collected data can be analyzed not only by device owners, but also by third parties.

At a theoretical level, questions such as "how can individuals challenge the use of their collected information ?", "how can companies express their privacy policies in a concise, meaningful and universal language ?" and "how can users give a valid form of consent to those privacy policies ?" are still open. To respect personal rights of individuals, answers to those questions shall include efforts from both the legal and technical sides. Note that, the ability to revoke earlier forms of consent has to be discussed as well.

Finally, if information security and privacy are threatened by the IoT, civil liability is also part of the new risks brought by this paradigm. Indeed, most connected devices can communicate between them to take decisions without requiring the physical intervention of their owners. Under those circumstances, the question "who is responsible if things go wrong ?" shall be addressed. In other words, when an IoT device causes damages, who will be responsible ? Are devices that were interacting or actors that were involved in the communications ? Furthermore, existing statutory civil liability regimes governing product liability and liability for physical injuries caused by an object are not suitable for the IoT. As the ins and outs are more complex than in other non-IoT cases, questions about civil liability thus need to be carefully studied.

To conclude, the development of the IoT has an evidence-based impact on the privacy of users. Data collected through those prevalent devices can be aggregated with other innovations to track activities and behaviors, but also to draw profiles of users including their traits of character as well as their medical conditions. As a consequence, it is urgent to combine legal efforts with technical ones in order to design privacy preserving features specific to the IoT, and answer the following questions: "what are techniques available to protect users ?", "how to impose their implementation ?", "how to verify that their implementation is correct ?" and "how to explain those mechanisms to the end-users ?".

II.7   Thesis motivation

In this section, we provide the motivation for this thesis with regard to the IoT related privacy issues, namely *physical tracking*, *device fingerprinting*, *activity inference* and *inventory attacks*. Facing the state of the art, both lacks of technical countermeasures and legal weaknesses are highlighted in the following.

Physical tracking is the process that exploits identifying elements in wireless communications in order to track IoT devices and their corresponding owners over time. In the literature, MAC address randomization has been introduced to tackle this threat. However,

this privacy preserving mechanism only focuses on a particular type of metadata: the device addresses. As a result, previous works [95, 77, 96, 81, 97, 1] demonstrated that this anti-tracking feature can be defeated leveraging other metadata such as sequence numbers included in Wi-Fi probe requests but also data embedded in wireless communications. Even though they can be protected thanks to encryption methods, this thesis pursues on the path started by those works asking the question: "how can those previously identified threats be generalized to other technologies (e.g. Bluetooth), and to the IoT ?".

Device fingerprinting offers another method of physical tracking based on the specific and unique configuration of IoT devices. To date, this threat has been studied a lot in the context of canvas [115] and browser fingerprinting [116], which ensure that companies can identify and track users across the web. Nevertheless, it has been demonstrated [77, 117] that data and metadata exposed in radio signals include elements that can be aggregated to form a device fingerprint. Even if it does not involve personal data per se, device fingerprinting allows to distinguish users, thus enabling the physical tracking. Compared to the existing research, this thesis aims to answer the question: "is fingerprinting also possible for IoT related wireless technologies (e.g. Bluetooth/BLE) ?".

Activity inference leverages series of observation to infer the activity of an individual. Assimilated to the activity inference, the Activity Recognition (AR) has been the subject of many works [118, 119, 120] where supervised machine learning techniques were used to mine data. Nonetheless, in the context of wireless communications, data broadcasted through radio signals are not always available in clear. As a consequence, to deal with the case where radio signals are the potential source of private leaks, this thesis will address the following questions: "in order to infer the activity of a user, is an approach based on the analysis of the temporal features of the wireless traffic possible to define ?" and, if so, "how can variations on the intensity of the wireless traffic (i.e. number of messages by unit of time) and on the periodicity of radio signals be leveraged ?".

Inventory attacks aim to infer private and potentially sensitive information based on a list of devices associated with a user. Given a list of device identifiers, this type of attack outputs personal attributes corresponding to the profile of the user. As a new threat raised in parallel with the deployment of the IoT devices, little work has been done. As a consequence, we will address this open research field through the design of an approach to infer information on users based on a list of their devices. To this purpose, this thesis will elucidate the following questions: "can a methodology be developed to associate personal attributes with a class of device ?" and, if so, "can it be automatized considering the information found in third party specifications and codes ?".

Legal regulations and protection authorities stipulate general laws that mostly apply to the collection and usage of personal data. Indeed, even if some efforts such as the extension dedicated to the IoT of the PIA framework from the CNIL [108] and the opinion of the G29 related to the deployment of the IoT [109] can be noticed, the case where wireless communications themselves can be sources of privacy leakages lacks from considerations. In this thesis, we are neither able to reform nor extend existing laws, and we do not pretend to do so. However, we hope that our scientific contributions

demonstrating privacy breaches through the IoT devices will bring evidences of legal weaknesses, thereby alerting regulators about the needs to design and harden new privacy protections specific to the IoT.

Consumer awareness is an act of making sure the consumer is aware of the information about the products, services and consumers rights. Even though they are more and more sensitive to privacy issues, consumers are generally not aware about privacy leakages through radio based attacks such as physical tracking and activity inference. To make the matter worst, they are even less aware that their personal information can be exploited by third parties for commercial or government intelligence purposes. Therefore, to raise awareness, we will develop demonstrators featuring some of the analyzed privacy threats. Enhanced with graphical interfaces allowing a clear exhibition of the results, such demonstrators will be showcased in scientific events to illustrate the conducted research, as well as in vulgarization events to address a wider audience (e.g the general public, data protection authorities and policy makers).

Chapter III

Tracking users leveraging advertising data and metadata

This chapter exposes how passive and active attackers can leverage advertising data and metadata broadcasted by wireless communicating devices in order to track their corresponding owners over time. To this end, physical tracking in the context of wireless communications is first discussed in Section III.1. Then, Section III.2 details the methodology used to collect and process two distinct datasets crowded with BLE advertisement packets and GATT profiles. Tracking concerns stemming from passive attacks are described in Section III.3, and Section III.3.5 presents Venom, a Visual and ExperimeNtal BluetOoth Low Energy tracking systeM developed for public awareness-raising purposes and experiments on privacy-enhancing features. GATT profiles fingerprinting, an active attack that can be also exploited to undermine the BLE privacy-preserving scheme, is studied in Section III.4. Section III.5 proposes Valkyrie (Verification of Addresses LinKabilitY in address Randomization ImplemEntations), a generic framework for verifying privacy provisions in wireless networks. Lastly, the conclusion for this chapter is provided in Section III.6.

Corresponding contributions : *[6, 9, 5, 4]*

III.1   Introduction

IoT devices are found in many applications and domains from healthcare, quantified self, entertainment as well as end-devices such as smartphones, tablets and laptop computers. All those battery-powered devices rely on wireless technologies such as Bluetooth/BLE or Wi-Fi to communicate. As a result, in their daily lives, users are carrying wireless enabled devices.

Because of the ever active discovery mechanisms, those devices periodically emit messages

that are populated with a variety of information including identifiers such as *device addresses* for Bluetooth and *MAC addresses* for Wi-Fi. Actually, those messages are broadcasted in clear, exposing their contents to eavesdroppers in range.

As a consequence, while wireless technologies bring hands-on facilities, they also have the potential to expose users to privacy threats such as physical tracking [121, 83, 67, 122, 59, 123, 124]. Sets of sensors are leveraged by tracking systems and collect such identifiers contained in signals emitted by wireless enabled devices. Afterwards, those identifiers are processed to detect the presence of users and estimate their mobility (see Section II.3). From customers analytics in shops [94, 125] to locating war opponents [126], through commuters monitoring [127] and urban planning [128], wireless based physical tracking has thus found diverse applications.

In response to this growing privacy concern, it has been proposed to replace those permanent identifiers with periodically changing random pseudonyms [85]. This practice, called *address randomization*, has been adopted by vendors [77, 81] and has even made its way to wireless standards [91, 38]. As an example, the Bluetooth Core Specification version 4.0 introduced the LE Privacy feature [39, Vol 3, Part C, sec. 10.7] that defines the use of temporary and random link layer identifiers. In addition, address randomization has become a default anti-tracking feature included in major mobile operating systems [129, 130].

In this chapter, we present an analysis of the privacy provisions of the LE Privacy feature, as well as its limitations. Our studies are based on two datasets of real-world observations of BLE advertisement packets and GATT profiles collected over a period of five months, and respectively associated with more than 53500 and 13200 different device addresses.

First, we analyze the deployment of the address randomization mechanism in the wild. Our results show that the LE Privacy feature is widely adopted and that most implementations follow the specifications by using short lived and uniformly distributed pseudonyms. Nevertheless, we also discovered that a still significant fraction of devices are not exhibiting those properties, exposing their users to tracking.

Second, despite the correct use of random device addresses, we demonstrate that the content of advertisement packets can defeat the LE Privacy feature. Indeed, we found that some devices include *static identifiers* in those packets and others include *counters* that are not reset upon an address change. Furthermore, we identified fields containing temporary identifiers, but for which the renewal is not done at the same time as the random address change. In particular, we point out that custom data included by *Apple*, *Microsoft* and *Google* proximity protocols can be leveraged to defeat the privacy-preserving scheme.

In parallel, it is important to mention that previous works have independently focused on the privacy aspects of BLE. Indeed, Issoufaly and Tournoux introduced [123] a BLE botnet of smartphones for wide scale tracking, and highlighted that BLE privacy features are not used by some wearable devices. In turn, Fawaz et al. presented *BLE-Guardian* [59], a solution to protect BLE users from privacy threats. In addition, they discussed several privacy issues of BLE advertisement, some of which are analyzed in detail in our research. More precisely,

they noted that some manufacturers do not use random addresses, and that some of the temporary identifiers are used for long periods of time. By providing an updated as well as a detailed review of those issues [1], we consolidate those observations. Likewise, Becker et al. demonstrated [97] that the LE Privacy feature can be defeated leveraging the advertising payload. By deeply investigating the device address randomization scheme fact, we complement this work not limiting us to empirical observations.

In order to raise public awareness toward physical tracking systems, we then follow our work by implementing *Venom*, a *Visual and ExperimeNtal BluetOoth Low Energy tracking systeM*. Used for demonstrational purposes, this system tracks users through their BLE enabled devices (e.g. headphones, smartwatches and fitness trackers), and displays collected data of participants thanks to its user-friendly interface. Note that, we also put an emphasis on the fact that *Venom* can be used to deploy and test privacy-enhancing features for physical tracking systems.

Last but not least, we focus on the GATT profile exposed by connectable BLE devices as part of the mandatory ATT protocol. As a reminder, this profile presents a description of features supported by a device through concepts of services and characteristics. Moreover, as most of its elements are readable without authentication, a GATT profile can be easily collected by any device in range. Leveraging the content of such a profile, we demonstrate that an *active* attacker can build a fingerprint that can be used to single-out the device and, once more, undermine the LE Privacy provisions.

Finally, we address the problem of verifying the correctness of the address randomization implementation to improve its effectiveness. More specifically, the objective is to check whether an implementation is affected by one or several previously identified issues (i.e. static identifiers and non-reset counters). Thereafter, we strengthen the privacy properties required by the address randomization that have been produced in recent research efforts.

To this purpose, we begin by formalizing the concept of frame unlinkability (that is the objective sought by address randomization) before to present *Valkyrie* (*Verification of Addresses LinKabilitY in address Randomization ImplemEntations*), a generic framework to automatically verify privacy provisions based on a network capture. Using a representative set of BLE and Wi-Fi enabled devices, we evaluate *Valkyrie* and conclude on the fact that this tool is able to detect issues in the generated wireless traffic.

III.2 Methodology and datasets

In this section, we present the methodology used to study tracking issues raised by the wireless communicating BLE devices. More particularly, we detail the approaches used to collect, parse, anonymize and sanitize our datasets. In order to have a complete view on

---

1. Several of the privacy concerns affecting BLE and discussed in this chapter (i.e. *static identifiers* and *non-reset counters*) were also found in the context of 802.11 [77, 81].

physical tracking concerns in the BLE, we studied this privacy threat triggering *passive* and *active* attacks. As a consequence, our work are based on two distinct datasets that we collected over five months during commute, work and leisure times. In the following, the dataset leveraged for the passive attacks is denoted as $dataset_{Passive}$, while $dataset_{Active}$ is the dataset used for the active attacks.

III.2.1   Data collection protocol

To collect the data, we used a Raspberry Pi [131] single-board computer equipped with several CSR v4.0 Bluetooth USB dongles [132].

On the one hand, $dataset_{Passive}$ is made of advertisement packets and scan responses. To collect advertisement packets on BLE advertising channels, we developed a C software based on the `BlueZ` [133] libraries, the official Linux Bluetooth stack. Furthermore, as advertising data can be obtained from both advertisement packets and scan responses (see Section I.2.2.3.1), we designed our software to collect advertisement packets and automatically send scan requests to obtain potential scan responses from discovered Peripherals.

On the other hand, $dataset_{Active}$ is populated with BLE GATT profiles. In order to collect those GATT profiles, one of the CSR Bluetooth dongles continuously scans for advertising Peripheral using the `bluepy` [134] Python library. The remaining dongles try to connect to discovered connectable Peripheral prior to enumerate attributes of their GATT profiles using our custom multi-threaded version of the `bleah` [135] BLE scanner tool.

III.2.2   Data structure

$dataset_{Passive}$ and $dataset_{Active}$ contain records that each respectively corresponds to an advertising frame (i.e. an advertisement packet or a scan response) or a GATT profile. A record includes a timestamp along with a header and a payload. The data obtained from the data collection are stored as raw records that are organized as follows:

— **Metadata**: timestamp;
— **Header**: PDU type, BD_ADDR type;
— **Payload**: BD_ADDR, advertising data ($dataset_{Passive}$) or GATT profile ($dataset_{Active}$);

Advertising data within $dataset_{Passive}$ are then parsed to extract AD structures and their various fields. To perform this task, we implemented our own Python parser based on online official Bluetooth SIG resources and other sources of information such as the `RaMBLE` [136] Android mobile application, the `advlib` [137] advertisement packet decoding library and `bleah`. The resulting data are then stored in a relational database which comprises 179

attributes corresponding to the metadata, fields of the header and parsed AD structures.

Concerning $dataset_{Active}$, the `bleah` tool is able to enumerate and parse a GATT profile by default. As a reminder, a GATT profile is a hierarchical structure of attributes that are grouped into *services*, each service containing conceptually related *characteristics* (see Section I.2.2.4). Therefore, we only formatted each captured GATT profile as a JSON string (see Figure B.1 in Appendix B) before to be stored in separate files for further analysis.

III.2.3   Datasets anonymization

Data collection of both $dataset_{Passive}$ and $dataset_{Active}$ were performed in the wild, from different items among which most of those are related to physical users. As a result, to minimize the privacy risks associated with the collected data, we applied modifications to prevent user re-identification. In particular, we focused on attributes corresponding to identifiers (Stable device addresses, device names, etc.) and on temporal data (timestamps). Therefore, the following transformations were applied:

— **Device addresses**: the Stable BD_ADDR (Public and Random Static) have been pseudonymized through keyed-hashing [1] of the 24-bit lowest part (NIC), thus leaving the 24-bit highest part (OUI) unmodified;
— **Names of persons**: fields potentially containing names were sanitized by searching and key-hashing substrings that were matching the pattern of a name (`"*.'s .*|*. 's .*"`, `"*.  de .*"`) or strings from a dictionary [2] of names;
— **Timestamps**: the temporal information has been transformed from absolute (date and time) to relative (time elapsed since the beginning of the collection campaign).

Nevertheless, to conduct the analysis of the random addresses distribution in $dataset_{Passive}$ (Section III.3.3.2), we kept the list of raw Random Static addresses that were stripped from any other information.

III.2.4   Datasets sanitization and preprocessing

Before performing our studies, $dataset_{Passive}$ as well as $dataset_{Active}$ went through steps of sanitization and preprocessing to remove unwanted records. Indeed, as our studies focus on physical tracking, we are only interested in BLE devices linked to individuals. However, there is a number of devices that does not fall into this category.

Filtering BLE beacons: BLE beacons are static devices deployed in the physical space to enable localization services such as indoor positioning, as an example. By definition,

---

1. The key used during this process has been erased.
2. We leveraged the online french last names [138] and first names [139] databases of INSEE.

Table III.1 – Beacon filtering regular expressions statistics.

Beacon type	Advertising data regular expressions	Adv. packets		BD_ADDR	
		%	#	%	#
Apple iBeacon	`^0201061aff4c000215`	0.63	50328	0.41	222
Google Eddystone	`^0201060303aafe.{2}16aafe[01234]0`	0.001	90	0.04	24
Radius Networks AltBeacon	`^1bff.{4}beac`	0.0003	21	0.006	3

they are not associated with an individual and are thus of little interest in the context of our studies. Based on regular expressions (see Table III.1) used to identify the main standards [1] of BLE beacons, the records associated with beacons have been filtered out the dataset. Using this approach, we identified and removed a total of 50439 advertisement records accounting for 0.63% of our original $dataset_{Passive}$. Note that, statistics for $dataset_{Active}$ are not available as we designed our software not to enumerate GATT profiles belonging to BLE beacons. In fact, leveraging the regular expressions on the broadcasted advertising data, our collecting device did not to connect to BLE beacons, thus ousting their corresponding GATT profiles from $dataset_{Active}$.

Filtering malformed payloads: Corrupted BLE frames are automatically discarded by the BLE stack thanks to the CRC. However, even if an advertisement packet has a correct CRC, it is still possible that some advertising data contain one or more malformed AD structures that can affect the correctness of our studies. We define a malformed AD structure as a structure that does not follow the format shown in Figure I.5, and thus cannot be correctly parsed. As a result, we removed all the records that have at least one malformed AD structure, leading to the suppression of 1958 advertisement records representing 0.02% of our original $dataset_{Passive}$. Similarly to the filtering of BLE beacons, statistics about malformed GATT profiles are not available as they were automatically removed during the enumeration process, leaving the $dataset_{Active}$ with only well-structured profiles. Note that, a malformed GATT profile is a profile that is not structured accordingly to Figure I.6 and, therefore, that cannot be further formatted as a JSON string.

To finish, Table III.2 presents the distribution of BD_ADDR and records among the device address types in our sanitized datasets. $dataset_{Passive}$ contains about 8 million records and includes more than 53500 distinct BD_ADDR, while $dataset_{Active}$ gathers GATT profiles from 13295 distinct BD_ADDR. Note that, devices using Private addresses (Random Non-resolvable and Random Resolvable) can be observed several times under a different pseudonym. As a consequence, in the Private part of the datasets, the number of actual devices is expected to be smaller than the reported number of distinct device addresses.

---

1. *Apple* iBeacon [140], *Google* Eddystone [141] and *Radius Networks* AltBeacon [142].

Table III.2 – Distribution of BD_ADDR and records among the device address types.

		Stable		Private	
		Public	Static	Non-res.	Res.
$dataset_{Passive}$	#BD_ADDR	4.4k	1.1k	9.5k	38.5k
	#Records	2.9M	220k	740k	4M
	#Advertisement packets	2.2M	130k	700k	2.6M
	#Scan responses	700k	90k	40k	1.4M
$dataset_{Active}$	#BD_ADDR	1.4k	0.5k	0.02k	11.4k

III.3 Passive tracking

In this work, leveraging $dataset_{Passive}$, we analyze the privacy issues associated with the advertising mechanism of BLE. First, we identify that, while widely adopted, the current implementation of BLE address randomization is suffering from serious issues. Indeed, some implementations fail at following the Bluetooth Core Specification on the maximum lifetime and the uniform distribution of random identifiers. Furthermore, we found that the payload of the advertisement packet can hamper the randomization mechanism by exposing static identifiers and non-reset counters. In particular, we discovered that custom data embedded in those frames by protocols of *Apple*, *Microsoft* and *Google* can be used to defeat the address randomization scheme.

Then, we present *Venom*, an experimental tracking platform aiming to raise public awareness about physical tracking technologies and to experiment privacy-preserving mechanisms. By collecting advertisement packets broadcasted by their BLE enabled devices, *Venom* tracks users and displays related information.

III.3.1 Attacker model

In this section, we consider an external attacker which continuously monitors the traffic on BLE advertising channels. This attacker is thus able to capture the advertisement packets generated by BLE devices in range. Moreover, we assume that the attacker is *passive*: he only captures traffic and does not interact with the wireless channel through injection or jamming.

On the victim side, we only make the assumption that his device has its Bluetooth interface turned on and broadcasts advertisement packets.

Based on the captured advertising traffic, the goal of the attacker is to track a device beyond the address randomization scheme. More specifically, a successful attack is defined as the attacker being able to link two sets of advertisement packets generated by a single device but with two distinct BD_ADDR.

Figure III.1 – Representation of a successful attack. The BLE device is randomizing its BD_ADDR (in *italic*) over time while keeping a static identifier (in **bold**) in the advertising data. Such a 5-byte identifier can be used by a *passive* attacker to link together the packets generated with the three different BD_ADDR.

Consequently, a successful attack links together two *pseudonyms* of a device. Figure III.1 shows an example where a device generates traffic with three different BD_ADDR, but it is possible to link together the three sets of advertisement packets based on a static identifier found within the advertising data.

III.3.2   Adoption of the LE Privacy feature

As a reminder, the device address is the main source of passive tracking attacks that have been addressed by the address randomization countermeasure (see Section II.5).

From Table III.2, we can first observe that Private addresses (Random Resolvable and Random Non-resolvable) account for about 60% of the advertisement packets found in $dataset_{Passive}$. This shows that the privacy-preserving mechanism of BLE is widely adopted by the industry. Also, this means that a number of devices are still using Stable addresses. In fact, $dataset_{Passive}$ includes more than 5500 Stable device addresses that can be trivially used to track the corresponding users [143, 59, 123]. Moreover, we computed that 80% of those Stable identifiers are of type Public, thus revealing the globally unique MAC address assigned by the manufacturer to the device.

Among the Private category, we can observe that a large part of advertisement packets belongs to the Random Resolvable type (4M packets) representing more than 84% of the traffic generated by Private addresses. As such, this suggests that the *resolvable* feature of LE Privacy is being endorsed by vendors over the *non-resolvable* option.

III.3.3   Device address randomization scheme analysis

In this section, we leverage $dataset_{Passive}$ to analyze current implementations of the LE Privacy feature, and more particularly the lifetime as well as the uniform distribution of the random addresses. Finally, the robustness of Random Resolvable addresses against a basic attack is analyzed too.

III.3.3.1 Lifetimes of device addresses

In order to prevent tracking, Random Resolvable and Random Non-resolvable addresses are temporary identifiers that are supposed to be renewed on a regular basis. In the Bluetooth Core Specification [38, Vol 6, Part B, sec. 6.1], a maximum duration of 15 minutes is recommended for both Random Resolvable and Random Non-resolvable addresses. On the opposite, Random Static addresses (the third type of random addresses) are not supposed to be changed, and are thus expected to be used for an extended duration.

Within this study, we computed the lifetime of each address found within $dataset_{Passive}$. Due to our method of data collection, the duration for which a device stays in range during the collection cannot be controlled nor measured. As a consequence, measured lifetimes should be considered as a conservative estimation. Indeed, a device may continue to use its BD_ADDR after [1] it goes out of the range of our collecting device.

For the four types of BD_ADDR, the cumulative distribution of the lifetimes of device addresses is presented in Figure III.2. First, we can observe that a significant fraction of Public and Random Static addresses have a long lifetime: 41% of Public and 17% of Random Static BD_ADDR have a lifetime of more than three hours. The very short lifetime of the remaining Stable addresses is likely due to the fact that the device was only observed during a single and short period of time.

On the other hand, we can observe that a vast majority of Private addresses have a very short lifetime as 89% of Random Resolvable and 96% of Random Non-resolvable addresses are observed for a duration below one second. Also, we can observe that a fraction of Private identifiers have a lifetime larger than 15 minutes. More specifically, 6% of Random Resolvable and 4% of Random Non-resolvable addresses exceed this limit [2].

Overall, those observations show that the recommendation of the lifetime of a device address is enforced by most implementations. However, some implementations of Private addresses are not following the specifications, keeping the same address for an extended duration and exposing the users to tracking.

III.3.3.2 Uniformity of random addresses distribution

According to the Bluetooth Core Specification [38, Vol 2, Part H, sec. 2], random addresses should be generated using a Federal Information Processing Standards (FIPS) compliant pseudorandom number generator (PRNG), and therefore should exhibit some characteristics expected from those PRNGs. In particular, it is expected that the output of those PRNGs follows a uniform distribution.

---

1. Likewise, we do not know how long the device used its BD_ADDR before to be caught by our collecting device.
2. Some Private addresses have even been observed for more than 69 days.
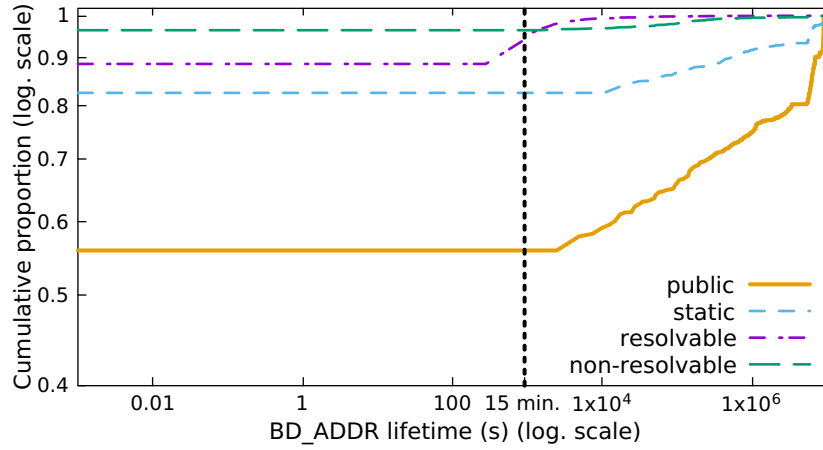
Figure III.2 – Empirical cumulative distribution function for lifetimes of device addresses.

For this research, we used the random addresses found within $dataset_{Passive}$ and tested whether this set of random values followed a uniform distribution[1]. For each random address, we extracted the part that has been generated with a PRNG[2] before to submit those values to the *Kolmogorov-Smirnov* (KS) statistical test [144].

By definition, the KS test is used to test whether one distribution of values differs substantially from theoretical expectations. It then outputs the distance (the *statistic*) between the empirical distribution and the reference distribution, as well as a significance score (the *p-value*) that gives the probability of obtaining this distance if the values were drawn from the reference distribution [145]. When the p-value is below a certain threshold (values of 0.1, 0.05, and 0.01 are typically used), the null hypothesis is rejected.

Run on each type of random addresses, this test produced the results reported in Table III.3. For Random Resolvable and Random Non-resolvable addresses, the p-values obtained are high (> 0.15), while the p-value for Random Static is very small ($< 10^{-5}$). As a result, this means that the hypothesis that all BD_ADDR are uniformly generated can be rejected for the Random Static addresses and kept for both Random Resolvable and Random Non-resolvable addresses.

Showing that some ranges of values are more common than others, the non-uniform distribution of Random Static addresses can be clearly observed on Figure III.3. In fact, we identified that some categories of devices are using specific values for the higher part of the device address (the leftmost 24 bits). For instance, *Sony* SRS portable speakers and *Samsung* Gear smartwatches respectively use `c7:d5:0b` (13 devices) and `dc:c1:c6` (10

---

1. Since those random values have been produced by different instances and implementations, it is thus not possible to test most of the properties required by FIPS.
2. Not all the bits of random BD_ADDR are supposed to be random. First, the two MSB are fixed to specify the type of the random address, and are consequently not random. Then, for Random Non-resolvable and Random Static addresses, the 46 remaining bits are random. Actually, for the Random Resolvable addresses, only 22 bits are random because the 24 LSB are computed from the random part `prand` and the IRK (see Section I.2.2.2).

Table III.3 – Results of the *Kolmogorov-Smirnov* test on the random part of random addresses against a uniform distribution.

BD_ADDR type	Statistic	p-value
Random Static	0.049615	3.851678×10^{-6}
Random Non-resolvable	0.006928	0.153595
Random Resolvable	0.002908	0.636685
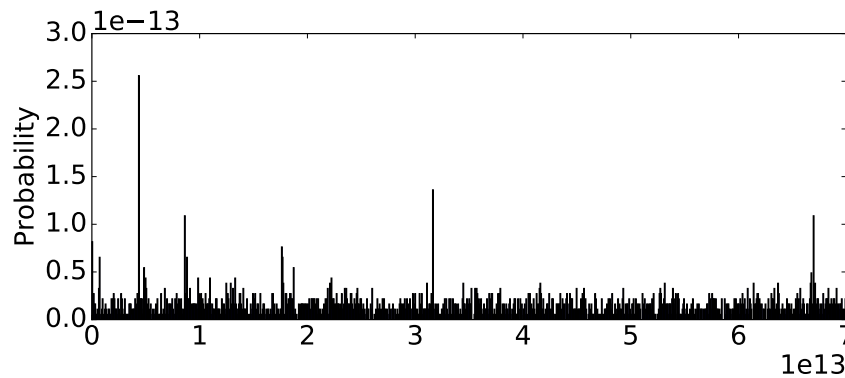
Figure III.3 – Distribution of Random Static addresses using a resolution of 1024 bins (size of a bin $= 2^{36}$).

devices) as a prefix. The same way MAC addresses are allocated by OUI, this suggests that some Random Static addresses are allocated by range, potentially revealing information on the type and manufacturer of the device (see Section IV.3.2).

Although the sets of Random Non-resolvable and Random Resolvable addresses appear to be uniformly distributed, this does not guarantee that all those random addresses are generated with a FIPS compliant PRNG, and that there are no other flaws in their generation.

III.3.3.3 Unsuccessful IRK brute forcing

To allow the identification of a Peripheral by a Central device with which it has already been paired (see Section I.2.2.2), Random Resolvable addresses include a non-random part. Actually, the 24 MSB of a Random Resolvable BD_ADDR are composed of two fixed bits specifying the type of the random address (Resolvable), followed by a 22-bit random part (`prand`), while the 24 LSB are equal to the hash of `prand` concatenated with a 128-bit secret value called IRK. As a result, the security of this scheme relies on the fact that the IRK cannot be easily guessed by an attacker.

Using the Random Resolvable addresses at our disposal, we tried a brute force attack using a dictionary containing *naive* values such as all zeros, all ones, incremental sequences

Table III.4 – List of IRK values used during the brute force attack. `hexstring` is defined as `0x0123456789abcdef`.

Description	IRK values	#Distinct IRK values
IRK furnished in the example of the Bluetooth Core Specification [38, Vol. 3, Part H, App. D.7]	`ec0234a357c8ad05341010a60a397d9b`	1
Naive incremental sequence of bytes	`0102030405060708090a0b0c0d0e0f10`	1
Naive sequence of bytes	`00112233445566778899aabbccddeeff`	1
Concatenation of two `hexstring`	`0123456789abcdef0123456789abcdef`	1
Same hexadecimal character	`00000000000000000000000000000000`, `11111111111111111111111111111111`, ...	16
`hexstring` interleaved with same hexadecimal character	`00102030405060708090a0b0c0d0e0f0`, `01112131415161718191a1b1c1d1e1f1`, ...	16
Same hexadecimal character interleaved with `hexstring`	`0001020304050607008090a0b0c0d0e0f`, `10111213141516171881891a1b1c1d1e1f`, ...	16
`hexstring` concatenated with same hexadecimal character	`0123456789abcdef0000000000000000`, `0123456789abcdef1111111111111111`, ...	16
Same hexadecimal character concatenated with `hexstring`	`00000000000000000123456789abcdef`, `11111111111111110123456789abcdef`, ...	16
Previous IRK values (Bluetooth example IRK, naive sequence of bytes, etc.) reversed by character	`b9d793a06a01014350da8c753a4320ce`, `01f0e0d0c0b0a0908070605040302010`, ...	84
Previous IRK values (Bluetooth example IRK, naive sequence of bytes, etc.) reversed by byte	`9b7d390aa610103405adc857a33402ec`, `100f0e0d0c0b0a090807060504030201`, ...	84
	Total	187

of numbers, as well as the IRK furnished in the example of the Bluetooth Core Specification [38, Vol 3, Part H, App. D7] (see Table III.4 for the detail of the dictionary). In total, we then tested 187 distinct IRK values against 38482 Random Resolvable addresses and found no matches. Therefore, this suggests that IRK used in real-world implementations are not naively generated.

III.3.4   Defeating device address randomization

According to the results of the previous sections, it appears that the address randomization implementation globally avoids common pitfalls. Nevertheless, in the following, we show that data such as static identifiers and non-reset counters included in the rest of the advertising payload can negate the privacy protection provided by the address randomization.

III.3.4.1   Static advertising identifiers

Most of the time, device names and service UUIDs are broadcasted by BLE enabled devices for identification purposes. In this section, we point out that such identifiers can undermine the anti-tracking feature when they are *static* (i.e. they do not change over time).

Device names: `Complete Local Name` and `Shortened Local Name` are two AD types that are respectively used to advertise a complete and a shortened version of local names assigned to devices. From $dataset_{Passive}$, we found that more than 1.7% of Random Resolvable and 0.06% of Random Non-resolvable addresses include either a `Complete Local Name` or a `Shortened Local Name` AD structure in their advertising payloads (see Table D.1 in Appendix D).

Moreover, we discovered that a number of device names advertised over Private addresses include a part corresponding to a unique identifier. For instance, device names of *Xiaomi Amazfit* fitness trackers match the following pattern `Amazfit-XXXX`, where `XXXX` are hexadecimal digits. Likewise, we identified several other families of devices such as *Boosted Boards* skateboards and *Nokia/Withings* smartwatches that include an identifier in their device names as well. In practice, the size of such an identifier varies from one to four bytes, and can be used to track the device over time.

From our observations on the Public addresses, we found that the identifier included in the device name is often the lowest part of the device address. For devices using Private addresses, this suggests that the digital identifiers embedded within the device names can be a part of the device Public BD_ADDR.

Service UUIDs: Service UUIDs are identifiers that can be found in several AD structures such as the `Complete List of 16-bit Service Class UUIDs` or the `Service Data-16-bit UUID` structures. Such UUIDs can be advertised in a full (128 bits) or in a shortened version (16 or 32 bits). In fact, their main purpose is to expose services to nearby Centrals before establishing a connection, but they can be also used to exchange service data in a connectionless way.

In this context, a UUID is tailored to identify a service rather than a device. However, we observed that some 128-bit UUIDs are customized by manufacturers to include a complete Public address in the LSB, thus forming a device unique identifier. Within advertisement packets using Private addresses, we found a total of 180 distinct UUIDs. Leveraging the registered OUI of the included Public addresses along with the advertised device names, we discovered that 1) some of those UUIDs belong to the *Nokia/Withings* manufacturer and 2) others are broadcasted by Steel smartwatches (9 devices). As such, this practice introduces a secondary unique identifier that can be used to track the device over time.

LE Bluetooth Device Address: The `LE Bluetooth Device Address` [146, Part A, sec. 1.16] AD type can be used by a Peripheral to broadcast a local device address and its class

Table III.5 – Summary of the studied proximity protocols along with their benched elements and tracking sources.

Proximity protocol	Benched element		Tracking source
	Name	Size	
Apple Handoff	`IV`	2 bytes	Non-reset counter
Apple Nearby Info	`Auth Tag`	3 bytes	Static identifier
Microsoft CDP	`Device Hash`	19 bytes	Static identifier
Google Nearby	`Counter`	2 bytes	Non-reset counter

(i.e. Public or random). Nevertheless, the Bluetooth Core Specification [146, Part A, sec. 1] requires not to embed such an AD structure in the advertising data. Especially, this is important for devices that use random BD_ADDR. However, we found that a number of devices using random addresses are including this AD structure in their advertising data, and expose their Public device addresses. Leveraging the OUI of those advertised addresses, we identified that manufacturers such as *LG Innotek* and *Arcadyan* expose their Public BD_ADDR this way.

III.3.4.2 Proximity protocols

In advertisement packets, AD structures such as the `Manufacturer Specific Data` and the `Service Data-16-bit UUID` structures can be used to carry application specific data. In the following, we focus on four prevalent proximity protocols found within advertisement packets, and discuss on the fact that their broadcasted data can expose information that can reduce or even negate the address randomization mechanism. Our findings are summarized in Table III.5.

III.3.4.2.1 *Apple* Handoff and *Apple* Nearby Info

The *Apple* Handoff [147] and *Apple* Nearby Info protocols are part of the *Apple* Continuity features.

Introduced in the *Apple* iOS 8 and *Apple* OS X Yosemite operating systems, the *Apple* Handoff protocol allows users to switch from one *Apple* device to another and seamlessly continue an ongoing activity. For instance, a user who is reading an article on its *Apple* iPhone smartphone can move to its *Apple* MacBook laptop in range and automatically open the same webpage.

To enable this feature, *Apple* devices rely on *Apple* Handoff data carried by the `Manufacturer Specific Data` AD structure. The format of such an *Apple* Handoff structure is presented in Figure III.4. It starts with an identifier indicating the type of the advertising data, in this case *Apple* Handoff (`0x0c`), and includes several fields such as an `Initialization Vector` (`IV`), an `AES-GCM Auth Tag` and a 10-byte encrypted payload.
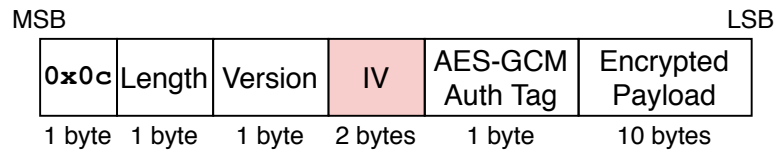
Figure III.4 – Format of the *Apple* Handoff `Manufacturer Specific Data` AD structure.

Time (s)	BD_ADDR	IV	Handoff Data
897.173	59:07:ee:1e:6c:72	fe02	2c2feab9d7...
898.594	59:07:ee:1e:6c:72	ff02	f8823f6a51...
899.256	59:07:ee:1e:6c:72	0003	1e4761159e...
899.820	59:07:ee:1e:6c:72	0103	a135354fb2...
900.003	76:46:5d:85:9e:f2	0103	a135354fb2...
900.252	76:46:5d:85:9e:f2	0203	0f4869c816...

Figure III.5 – Sequence of *Apple* Handoff advertisement packets showing that the IV is not reset after the change of BD_ADDR at 900.003.

Acting as a frame *counter*, the `IV` is a 2-byte integer that is incremented over time, but that can remain unchanged in several consecutive advertisement packets. Actually, we observed that this counter is not reset when the BD_ADDR changes in the randomization scheme (see Figure III.5). Based on this `IV`, it is then possible to link different addresses belonging to the same device, as it has been shown with the `Sequence Number` field included in 802.11 probe requests [77].

On the other hand, although it is not documented, the *Apple* Nearby Info protocol seems to be used to inform nearby devices about the presence and state of a device (see Section V.2.4). Note that, in this work, we only focus on two particular *Apple* Continuity protocols (i.e. *Apple* Nearby Info and *Apple* Handoff). Nevertheless, in Chapter V, we push this study further by extending our results to the suite of *Apple* Continuity protocols.

Similarly to the *Apple* Handoff, the *Apple* Nearby Info protocol relies on the `Manufacturer Specific Data` AD structure to broadcast data. Following the same type-length-value (TLV) encoding scheme, the *Apple* Nearby Info AD structure starts with an identifier indicating the *Apple* Nearby Info protocol (`0x10`). Afterwards, it includes several fields such as the `Activity Level` of the device and an `Auth Tag` (see Figure III.6).
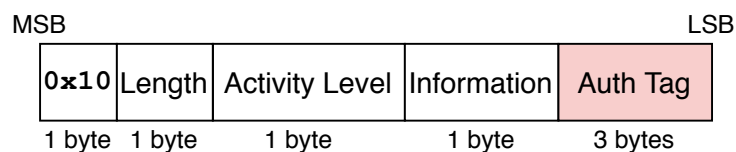
Figure III.6 – Format of the *Apple* Nearby Info `Manufacturer Specific Data` AD structure.

Time (s)	BD_ADDR	Auth Tag
899.885	43:26:33:d5:78:61	60c708
899.990	43:26:33:d5:78:61	60c708
900.091	6d:01:ff:0a:52:84	60c708
900.203	6d:01:ff:0a:52:84	9d88fb
900.354	6d:01:ff:0a:52:84	9d88fb

Figure III.7 – Sequence of *Apple* Nearby Info advertisement packets showing that the BD_ADDR and *Apple* Nearby Info `Auth Tag` changes are not synchronized. At 900.091, the old `Auth Tag` is used with the new BD_ADDR.
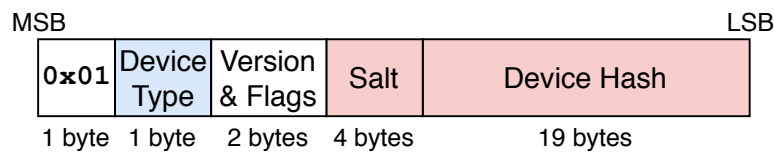
Figure III.8 – Format of the *Microsoft* CDP `Manufacturer Specific Data` AD structure.

To prevent physical tracking based on the `Auth Tag`, the *Apple* Nearby Info protocol periodically rotates this 3-byte identifier every time the BD_ADDR is changed. However, based on our observations on an *Apple* iPhone 6 and an *Apple* iPhone 8 smartphone (respectively running iOS 12.1.3 and iOS 12.3), the change of the `Auth Tag` is not completely synchronized with the change of the BD_ADDR: for a short duration after the BD_ADDR change, the old `Auth Tag` is used with the new device address (see Figure III.7). Leveraging the `Auth Tag`, it is then possible to link two BD_ADDR generated by the same device.

In $dataset_{Passive}$, we found that among the 31000 Private addresses that include an *Apple* `Manufacturer Specific Data` AD structure, more than 79% of them include an `Auth Tag`.

III.3.4.2.2   *Microsoft* Connected Devices Platform (CDP)

The *Microsoft* Connected Devices Platform (CDP) [148] protocol provides a discovery system for *Microsoft* devices to authenticate and verify themselves, as well as a way to exchange data. Similarly to the *Apple* Handoff and *Apple* Nearby Info protocols, data of *Microsoft* CDP are carried by the `Manufacturer Specific Data` AD structure.

Based on the online *Microsoft* specifications along with empirical observations on advertisement packets, we identified the format of the *Microsoft* CDP AD structure (see Figure III.8). In particular, it embeds an identifier that is derived from a salted hashing of the unique device identifier [148, sec. 2.2.2.2.3].

Using a BLE enabled laptop running *Microsoft* Windows 10 Professional (version 1809,

Time (s)	BD_ADDR	Device Hash
959.108	37:ee:cb:91:79:0a	db950efc53eff7e427f2a91ae9a67b...
959.522	37:ee:cb:91:79:0a	db950efc53eff7e427f2a91ae9a67b...
959.719	18:e3:48:43:af:84	db950efc53eff7e427f2a91ae9a67b...
1919.074	2d:39:47:eb:2c:e8	db950efc53eff7e427f2a91ae9a67b...
2879.527	19:fc:04:f1:f3:9a	db950efc53eff7e427f2a91ae9a67b...
3599.113	19:fc:04:f1:f3:9a	db950efc53eff7e427f2a91ae9a67b...
3599.189	19:fc:04:f1:f3:9a	4658a402b7da02e09585cb8c4aa1c7...
3839.851	19:fc:04:f1:f3:9a	4658a402b7da02e09585cb8c4aa1c7...
3839.984	39:95:ae:3f:ed:cb	4658a402b7da02e09585cb8c4aa1c7...

Figure III.9 – Sequence of *Microsoft* CDP advertisement packets in which the lifetime of the `Device Hash` overlaps the BD_ADDR randomization scheme. At 959.719, the BD_ADDR changes while the `Device Hash` remains identical until 3599.113.

OS Build 17763.437) left idled, we witnessed that the lifetime of the BD_ADDR is about 16 minutes, while the `Device Hash` lasts for approximately 60 minutes (see Figure III.9). First, we can observe that Random Non-resolvable addresses used to advertise *Microsoft* CDP data last longer than the Bluetooth Core Specification recommended duration of 15 minutes. Thereafter, the switch to a new `Device Hash` is not synchronized with the change of BD_ADDR. As a result, the *Microsoft* CDP advertising data undermine the address randomization scheme through the exposure of an identifier whose lifetime overlaps the one of the BD_ADDR.

From $dataset_{Passive}$, we computed that more than 89% of Random Non-resolvable addresses include the *Microsoft* company identifier (`0x0006`) in the `Manufacturer Specific Data` AD structure and follow the format of the data shown in Figure III.8.

Leveraging the `Device Type` field found within the *Microsoft* CDP data, we identified that 96% of such Random Non-resolvable addresses are broadcasted from *Microsoft* Windows 10 computers (desktops and laptops). Despite the use of address randomization, this suggests that a large fraction of the BLE enabled *Microsoft* devices are trackable. Furthermore, it appears that most *Microsoft* Windows 10 computers are affected by this issue as we discovered that `CDPUserSvc`, the service responsible for broadcasting those *Microsoft* CDP messages, is enabled by default.

III.3.4.2.3   *Google* Nearby

The *Google* Nearby [149] protocol is an Android feature that allows users to discover devices in range and establish direct communication channels with them. Moreover, this protocol is present on all modern Android devices since Android KitKat (version 4.4) via *Google* Play services [150].

Unlike the *Apple* Handoff, *Apple* Nearby Info and *Microsoft* CDP protocols, the *Google* Nearby data are carried by the `Service Data-16-bit UUID` AD structure. Relying on the online *Google* specifications [151] and codes [152] along with an analysis of BLE
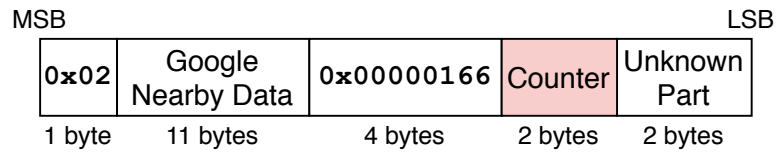
Figure III.10 − Format of the *Google* Nearby `Service Data-16-bit UUID` AD structure.

advertisement traffic, we partially identified the format of the *Google* Nearby payload (see Figure III.10). More precisely, this payload includes a 2-byte `Counter` that is incremented over time, but which does not necessarily change for each advertisement packet. In fact, as for the *Apple* Handoff protocol, we observed that this `Counter` is not reset when the device switches to a new random address. In addition, we found that a 3-byte static identifier is included in the `Manufacturer Specific Data` AD structure of the scan responses of the Peripheral, and can be also leveraged to defeat the randomization scheme.

In $dataset_{Passive}$, we computed that more than 17.7% of Random Resolvable addresses advertise a *Google* 16-bit service UUID over the `Service Data-16-bit UUID` AD structure, among which 39% of those respect the format of the *Google* Nearby service data presented in Figure III.10. Leveraging the advertised device names, we detected that some of those broadcasted service data come from *Google* Chromecast devices. Indeed, the *Google* Nearby protocol is used by such devices to facilitate their setup process [153].

Since *Google* Chromecast are usually sedentary devices, the tracking risk is limited. However, the *Google* Nearby protocol is designed to be integrated in any type of device, and especially mobile handsets and wearable equipment.

III.3.5 Practical application: *Venom*, an experimental BLE tracking system

Despite efforts from both the industry and data protection authorities, the previous sections demonstrated that the privacy of users is still in jeopardy. Additionally, the fact that tracking technologies are not well known by the general public aggravates the situation.

In this section, we then introduce *Venom*, a *Visual and ExperimeNtal BluetOoth Low Energy tracking systeM*, to shed light on privacy issues of BLE enabled devices. In the following, we show how it can be used for demonstrational purposes in order to raise user awareness about radio based physical tracking technologies. Also, we explain how this platform can be leveraged as a basis to develop and test privacy-preserving mechanisms. To this end, we present a primitive and easy-to-use BLE based opt-out mechanism that does not involve any action from users on their devices, unlike the one proposed in *Wombat* [154].

III.3.5.1 The *Venom* tracking system

Venom is a BLE tracking system supporting collection, storage and processing of advertisement packets. Those features are implemented over a distributed infrastructure composed of:

— **Clients**: wireless monitoring-capable devices that collect advertisement packets and forward them to the broker;
— **Broker**: receives data from clients, stores and processes them (i.e. parse advertisement packets, analyze advertising data, etc.) in a local database.

To collect advertisement packets, *Venom* only requires Bluetooth interfaces supporting the BLE protocol. Actually, this is the case for most off-the-shelf Bluetooth cards on computers running a Unix-like system. To enable the communication between the clients and the broker, the *Venom* tracking system relies on its own custom Wi-Fi network. Finally, the advertisement packets parser has been built from the online official Bluetooth SIG resources, third-party public resources[1] and the reverse engineering of the *Apple* Handoff, *Apple* Nearby Info, *Microsoft* CDP and *Google* Nearby proximity protocols.

In addition to its principal features, we enriched *Venom* with a user interface along with an opt-out client:

— **User interface**: displays information about tracked devices through proximity detection[2]. Note that, the interface has been made to address as much types of audiences as possible (i.e. the general public, industrials, researchers and students);
— **Opt-out client**: implements a basic opt-out mechanism for users that do not want to be tracked by the system (see Section III.3.5.2).

Figure III.11 presents the architecture of *Venom* that features core functionalities of industrial Bluetooth/BLE tracking systems: device detection, identification and itinerary tracking. To present principles of radio-based tracking systems, and to initiate discussions on corresponding privacy issues, we assume that those functionalities are enough.

Information captured by the system are only data contained in advertisement packets sent by BLE devices having an enabled Bluetooth interface. Thus, traffic data from associated devices and timing or physical-layer information are not considered. The display includes metadata of the advertisement packet (i.e. BD_ADDR, BD_ADDR type, etc.), and a list of extracted information along with an inference of the device type based on its advertised device name.

To minimize privacy risks for users, we keep as little necessary information as possible. In particular, collected data of a participant are kept for a maximum of 15 minutes after its

---

1. The `advlib` [137] advertisement packet decoding library, the `RaMBLE` [136] Android mobile application, and the `bleah` [135] BLE scanner tool.
2. Nearby devices are detected using the RSSI.
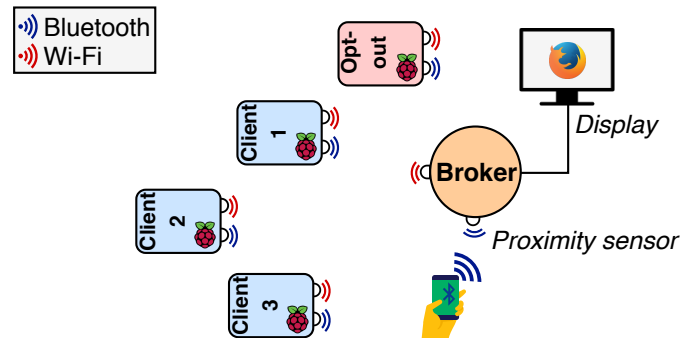
Figure III.11 – Architecture of the *Venom* tracking system.

departure. Furthermore, *Venom* only detects devices in close range[1] of antennas to ensure that only volunteering participants will have their data collected and processed.

III.3.5.2   Privacy protection feature

Most tracking systems collect data of users without their consents. Therefore, to bring users possibilities to escape tracking, opt-out mechanisms have been deployed. Generally, such mechanisms involve a webpage on which the user has to declare its device address. However, this approach presents usability issues preventing users from protecting their privacy through opt-out mechanisms.

In this work, we leverage BLE core elements to transmit the opt-out[2] decision. In fact, we implemented a usable opt-out mechanism to which a device, whose owner wants to opt out, only has to be close to the `Opt-out` Bluetooth antenna of *Venom*. Note that, this differs from the opt-out mechanism introduced in [154], with which Wi-Fi devices willing to opt out must associate to an access point. Finally, upon this event, the tracking system will learn the BD_ADDR by parsing received nearby advertisement packets.

III.3.5.3   Interaction with participants

During the demonstration, visitors are tracked leveraging advertisement packets broadcasted by their carried BLE devices. At the entry, they are informed of the presence of the tracking system and the opt-out (or opt-in) mechanism. By bringing their device close to the `Report` Bluetooth antenna of *Venom*, they are able to test the information collected from their devices. As a feedback to the participant, a comprehensive analysis of the collected data (i.e. identifier, brand of the device, etc.) along with an approximate

---

1. A few centimeters.
2. Depending on the demonstration purpose, *Venom* can be configured to either set an opt-out or opt-in mechanism.

representation of the user itinerary inside the room is then displayed on a screen (see Figure A.1 in Appendix A).

III.4   Active tracking based on GATT profiles

In the context of wireless technologies, the possibility of singling-out a device based on its technical characteristics and attributes for tracking purposes has been explored by several works [155, 97, 1]. Moreover, in [77, 81], the authors demonstrated that the content of 802.11 probe requests can be used to fingerprint devices and defeat the address randomization.

In the following, we give evidence that this problem is not limited to 802.11 as BLE suffers from similar issues. To this end, we discuss how the GATT profile – which is a hierarchical structure of attributes allowing the transfer of information between a Client and a Server (see Section I.2.2.4) – can be used to create a fingerprint that can be exploited to circumvent the anti-tracking feature of the BLE standard (i.e. the device address randomization). Leveraging $dataset_{Active}$, we analyze the potential of this fingerprint and show that it can be used to uniquely identify a number of devices.

III.4.1   Attacker model

In this section, we consider an *active* attacker which monitors the BLE advertising channels to detect nearby connectable Peripherals, connect to them and collect their GATT profiles using several ATT `Read By Type Request` [38, Vol 3, Part F, sec. 3.4.4]. Furthermore, we assume that the device used by the attacker has not been paired with any Peripheral: it cannot authenticate itself and access protected values of characteristics. As described in Section III.2.1, those assumptions can be satisfied using off-the-shelf hardware and open-source software. In addition, we found that a full GATT profile can be collected in a matter of seconds (see Table III.6).

On the target side, we assume that the device has its Bluetooth interface turned on, is in communication range and is discoverable.

Based on the collected GATT profile, the objective of the attacker is to generate a fingerprint of the device in order to track it despite its address randomization scheme (see Figure III.12).

Table III.6 – Average time to collect a GATT profile among different devices.

Type of device	Device	Time (s)
Lightbulb	*Osram* Smart+	6.531
Motion sensor	*Eve* Motion	6.468
Socket outlet	*Eve* Energy	5.919
Smartphone	*Apple* iPhone 8	4.354
Smartphone	*Apple* iPhone 6	4.259
Keyring	*Nut*	4.148
TV dongle	*Google* Chromecast	3.660
Fitness wristband	*Fitbit* Inspire	3.231
Presentation remote	*Logitech* Spotlight	2.860
Smartwatch	*Apple* Watch Series 3	2.853
Heart rate monitor	*Polar* H7	2.751
Fitness wristband	*Fitbit* Flex	2.552
Headset	*Bose* SoundLink Around-Ear II	2.181
Speaker	*Divacore* Ktulu2+	1.742
Keyring	*Chipolo*	1.426
Average		3.662

5b:eb:b9:c5:f4:ee   6c:dc:32:08:38:04   5f:c6:5c:25:b7:5a

Figure III.12 – Representation of a successful attack. The BLE device is randomizing its BD_ADDR (in *italic*) over time while keeping a static GATT profile. Such a GATT profile can be leveraged by an *active* attacker to generate a fingerprint of the device before to link together its three different BD_ADDR.

III.4.2   GATT profiles fingerprinting

Following the approach of Vanhoef et al. [77] that was applied to 802.11 probe requests, we study how much identifying information can be found in GATT profiles. Especially, we study how services and characteristics can be used to create a fingerprint of the device. In case this fingerprint is unique enough, it can be used to track a device despite the address randomization.

III.4.2.1 GATT fingerprint artifacts

The GATT profile of a BLE device is a data structure that can be easily accessed and that includes a number of data elements that can be used for fingerprinting.

First, the number of possible components is large: online GATT specifications [47] describe a list of 40 services and 226 characteristics that can be complemented by vendors with their own custom elements. In total, we found 263 distinct services and 1086 distinct characteristics in $dataset_{Active}$. In addition, characteristics are associated with a value that can contain up to 512 bytes of data [38, Vol 3, Part F, sec. 3.2.9]. Also, all those elements are accompanied by metadata: handles and properties respectively represented by two bytes and eight flags.

Second, the content of a profile will vary depending on the device type as the GATT profile reflects the features and the characteristics of the device. For instance, the `Cycling Power Measurement` characteristic will be only included in cycle devices, while only weight scales will expose the `Weight Scale Feature` characteristic.

Finally, values associated with characteristics may vary from one device to another as they can reflect the device state or identity. As an illustration, this is the case of identifiers such as the `Device Name` and `Model Number String`.

Overall, a GATT profile is a data structure containing a large number of elements that are subject to variations between devices, and thus hold a potential for fingerprinting. To create the fingerprint of a BLE device, we considered the following artifacts:

- List of *services*, including for each service:
  - **Handles (start-end)**: the handle range associated with the service (two 16-bit identifiers);
  - **UUID**: the UUID associated with the service (a 128-bit identifier).
- List of *characteristics*, including for each characteristic:
  - **Handle**: the handle associated with the characteristic (a 16-bit identifier);
  - **UUID**: the UUID associated with the characteristic (a 128-bit identifier);
  - **Properties**: the properties of the characteristic (8 bits);
  - **Value**: the value of the characteristic (from 0 to 512 bytes).

III.4.2.2 Fingerprinting evaluation

To evaluate the fingerprinting potential offered by the GATT profiles, we leveraged $dataset_{Active}$. Previously presented artifacts were then extracted from $dataset_{Active}$ prior to be stored in a database in which each fingerprint is associated with a device address and a timestamp. Thereafter, the resulting database was processed to compute fingerprinting metrics: entropy (Section III.4.2.3) and anonymity sets (Section III.4.2.4).

Impact of random addresses on the evaluation: $dataset_{Active}$ includes records from devices using random addresses (Private addresses). A device using the address randomization scheme can be observed multiple times under different pseudonyms, and thus the corresponding fingerprint will be counted multiple times instead of one. This overcounting will have an impact on the privacy metrics: the entropy will be reduced and the size of the anonymity set will be increased. Therefore, values reported for the Private part of $dataset_{Active}$ shall be considered as an underestimation of the fingerprinting potential.

III.4.2.3 Empirical entropy

Leveraging $dataset_{Active}$, we evaluate the quantity of information brought by the services and characteristics of GATT profiles. The entropy is a metric used to measure the amount of identifying information brought by an element of the fingerprint [116]. In this work, the database of fingerprints was processed to compute an empirical evaluation of the entropy of each artifact i using the following formula:

$$H_i = -\sum_{j \in E_i} f_{i,j} * \log_2 f_{i,j} \tag{III.1}$$

where E_i is the domain of possible values for an artifact i and $f_{i,j}$ is the frequency (i.e. probability) of the value j for the artifact i in $dataset_{Active}$. Note that, the absence of an artifact was also considered as a possible value.

Table III.7 presents the entropy for the ten most common services and characteristics exposed in $dataset_{Active}$, as well as for the overall profile. The *Entropy* column presents the amount of identifying bits provided by the artifacts. The *Stability* column presents the fraction of devices observed several times for which the value of the artifact is constant throughout $dataset_{Active}$. Finally, the *Affected devices* column presents the fraction of devices that include this artifact in their GATT profiles.

A first observation is the high stability of the fingerprint: the overall fingerprint is stable in more than 95% of the cases. The entropy of single artifacts is typically comprised between 0 and 2 bits. However, some artifacts such as the `Device Name` and the `Model Number String` characteristic can bring up to 3.152 bits of information. Indeed, those artifacts are in fact identifiers.

Variations can be also observed between the types of device address: the `Device Name` brings less information for Private than for Stable addresses. Actually, we witnessed that for devices using Private addresses, this characteristic is often configured to carry a generic value[1]. This is likely a deliberate choice done for privacy reasons. Nevertheless, developers

---

1. For instance, the value of the `Device Name` characteristic is `iPhone` for both an *Apple* iPhone 6 and an *Apple* iPhone 8 smartphone (see Figure B.1 in Appendix B).

Table III.7 – Empirical entropy computed from $dataset_{Active}$ for services and characteristics exposed within GATT profiles. For each item: the entropy brought by the attribute, the percentage of devices for which this item is stable over time, and the percentage of devices that include this item in their GATT profiles.

		Entropy (bits)			Stability* (%)			Affected devices (%)		
		All	Stable	Private	All	Stable	Private	All	Stable	Private
Services $\langle$Hdle Start, Hdle End, UUID$\rangle$	Generic Access	0.750	0.606	0.245	100	100	100	99.01	92.99	100
	Generic Attribute	0.728	0.560	0.248	100	100	100	97.34	81.35	99.97
	Apple Continuity	0.680	0.317	0.462	100	100	100	84.74	6.48	97.64
	Apple Nearby Service	0.720	0.306	0.499	100	100	100	84.32	4.36	97.50
	Device Information	1.425	0.551	1.037	100	100	100	69.74	55.90	72.02
	Battery Service	0.943	0.328	0.879	100	100	100	57.86	5.58	66.48
	Current Time Service	0.871	0.277	0.866	100	100	100	57.08	0.05	66.48
	Apple Media Service	0.835	0.277	0.831	100	–	100	57.07	0	66.48
	Apple NCS Service	0.843	0.277	0.838	100	–	100	57.07	0	66.48
	ISSC Transparent	0.170	0.366	0.131	100	100	–	3.86	27.26	0
	Overall	2.111	0.803	1.330	99.28	100	99.20	–	–	–
Characteristics $\langle$Hdle, UUID, Props, Value$\rangle$	Device Name	1.913	1.191	0.731	100	100	100	99.65	97.56	100
	Appearance	1.148	0.625	0.578	100	100	100	98.90	92.40	99.97
	Service Changed	0.766	0.566	0.290	100	100	100	97.34	81.35	99.97
	Apple Continuity	0.680	0.317	0.462	100	100	100	84.74	6.48	97.64
	Apple Nearby	0.720	0.306	0.499	100	100	100	84.32	4.36	97.50
	Manuf. Name String	1.422	0.538	1.053	99.82	100	99.81	69.38	53.40	72.02
	Model Number String	3.152	0.564	2.757	99.82	100	99.81	69.32	52.98	72.02
	Battery Level	1.020	0.395	0.879	100	100	100	58.65	11.16	66.48
	Current Time	0.893	0.297	0.866	100	96.77	100	57.30	1.65	66.48
	Apple MS Entity Att.	0.865	0.277	0.861	99.58	–	99.58	57.07	0	66.48
	Overall	4.380	1.294	3.092	97.84	95.71	98.08	–	–	–
	Overall (services + characteristics)	4.380	1.294	3.092	97.84	95.71	98.08	–	–	–

* Stability values have been computed only from device addresses that we observed multiple times.

appear to have overlooked the `Model Number String` as it appears to be a high source of information for Private addresses (2.757 bits).

Overall, characteristics seem to bring more information than services (4.380 bits against 2.111 bits). This is explained by the fact that characteristics hold more artifacts than services. When considering the full fingerprint, which includes both the characteristics and the services, we can observe that the entropy is the same as with the characteristics alone. This is due to the fact that artifacts of a service (i.e. handles and UUID) are fully determined by artifacts of its characteristics (remind that characteristics are hierarchically dependent from services). In other words, services do not bring additional information with regard to characteristics.

(a) Stable addresses
(1882 devices)

(b) Private addresses
(11413 devices)

(c) Private without
iPhones (1489 devices)

(d) Private iPhones only
(9924 devices)
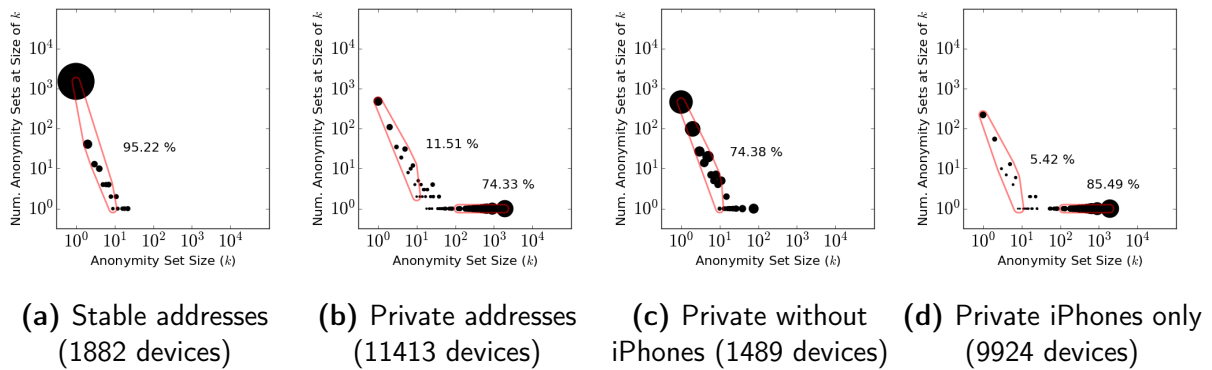
Figure III.13 – Anonymity sets of GATT profiles in $dataset_{Active}$. The dot size is proportional to the number of devices in the set.

III.4.2.4   Anonymity sets

To further study the fingerprinting potential of GATT profiles, we used the concept of *anonymity set*, which is defined as a set of entities that share the same fingerprint. From a privacy point of view, the larger the anonymity set the better.

Aided by the `kmap` [156] Python tool, we computed the anonymity sets for the fingerprints contained in $dataset_{Active}$. Figures III.13a and III.13b respectively show the distributions of the set sizes for Stable and Private addresses.

For Stable addresses, the anonymity sets are small with 94.75% of sets of size 1, meaning that those devices can be uniquely identified by their fingerprints. As those devices can be already identified through their Stable addresses, this is not critical. However, it demonstrates the potential for unique identification based on the GATT profile.

Moving to Private addresses, we observe that a smaller number of devices are uniquely identifiable (4.28%), and that 74.33% of devices are in anonymity sets of size 100 or more. This improvement can be explained by the fact that vendors include less identifying information in GATT profiles of devices using Private addresses.

Focusing on devices using Private addresses, we found that a large number of them are *Apple* iPhone smartphones[1]. By dividing the Private part of $dataset_{Active}$ between non-iPhone devices (see Figure III.13c) and iPhones (see Figure III.13d), we found that a majority of iPhones were sharing their fingerprints with many other devices: 85.49% are in anonymity sets of size 100 or more. On the other hand, non-iPhone devices using Private addresses have less common fingerprints as 74.38% of them are in anonymity sets of size 10 or less, and 32.09% of them are unique.

A possible explanation to this phenomenon is that *Apple* distributes a large number of

---

1. The identification of the device model is based on the values of the `Model Number String` and `Manufacturer Name String` characteristics along with the presence of *Apple* specific services (`Apple Continuity Service`, `Apple Nearby Service`, etc.).

devices but focused on a small number of models (i.e. a single line of products with few variants per generation). Furthermore, the software running on those devices is homogeneous[1]. As such, it seems that a side effect of *Apple* commercial and technical policies is to reduce possibilities of uniquely identifying their devices based on technical characteristics.

III.5 Verification of privacy provisions in wireless networks

With regard to the tracking issues detailed in the previous sections, the correct implementation of the device address randomization appears not to be trivial.

In this section, we thus address the problem of verifying the correctness of an address randomization implementation through an automated verification tool. To this end, we introduce an approach to identify issues based on a capture of the traffic generated by a device. In fact, this approach relies on rules specifying requirements for a correct implementation of address randomization.

Then, we prototype *Valkyrie* (*Verification of Addresses LinKabilitY in address Randomization ImplemEntations*), a software tool that, based on a set of rules, verifies that a given sequence of frames generated by a device does not compromise the address randomization scheme.

Finally, we evaluate this tool on a corpus of frame captures corresponding to 60 devices implementing address randomization for both BLE and Wi-Fi. As a side note, we highlight that the results are convincing as *Valkyrie* is able to detect the previously identified tracking issues (i.e. static identifiers and non-reset counters) in the generated wireless traffic.

III.5.1 Address randomization and its limitations

Following the appearance of wireless tracking [158, 159, 67, 160, 161, 122], address randomization has been introduced to protect privacy of users. As a reminder, the address randomization idea is to replace the link layer identifier[2] with a temporary and random one. This countermeasure denies the link layer identifier to be used as a reliable element for tracking.

In Wi-Fi, address randomization has been adopted in various OS (i.e. iOS, Android, Windows and Linux), and recently in the 802.11 standard [91]. In Bluetooth, address

---

1. *Apple* devices regularly receive software updates to ensure maximum security and functionality [157].
2. In general, a globally unique identifier.

4e:a5:8a:66:18:2d *6b:41:bc:1b:9e:d5* *5c:d0:7c:6b:c0:7c*

021f**22c5**e7fab5 14e8**22c6**a87bd1 8a6b**22c7**215fa9
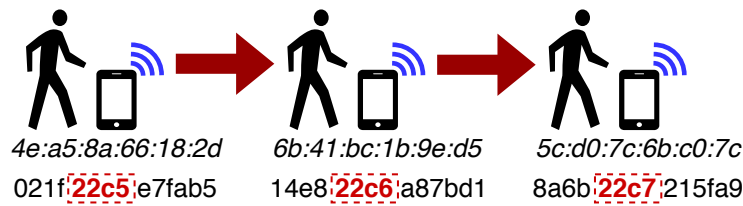
Figure III.14 – Example of the device address linking via a non-reset `Sequence Number` field. The device is randomizing its address (in *italic*) over time while incrementing the `Sequence Number` (in **bold**) in the broadcasted data. Such a 2-byte long counter can be leveraged by a *passive* attacker to link together frames generated with the three different device addresses.

randomization was introduced in 2010 through the version 4.0 of the standard [39, Vol 3, Part C, sec. 10.7]. Moreover, according to Becker et al. [97] and our observations in Section III.3.2, it seems that address randomization is included in a significant part of BLE devices.

Despite a large adoption of this anti-tracking measure, we previously showed that using a rotating link layer identifier is not enough to prevent tracking. More precisely, the rest of the frame may include other unique identifiers or artifacts that can be used for tracking (see Section III.3) or fingerprinting (see Section III.4) a device over address changes [77, 81].

For instance, a counter (i.e. `Sequence Number`) included in 802.11 frames was not reset upon the address change in the early randomization implementation in iOS [95]. As a consequence, it was possible to link together two consecutive address fields by observing the increasing values of the `Sequence Number` field (see Figure III.14).

Even if the link layer identifier is correctly rotated, overlooked elements and implementation errors can then still undermine the privacy protection.

III.5.2 Privacy properties of network traffic

In this section, we discuss properties necessary to prevent tracking in face of a *passive* attacker.

III.5.2.1 Frame unlinkability

The objective of measures such as address randomization is to avoid an observer from tracking a device over an extended period of time. To achieve this objective, we argue that it is mandatory to prevent the attacker from linking together frames generated by a single device.

Frame linking can be done based on their contents [77, 81], timings [96] or even their properties at the physical layer [76]. In this section, we only focus on the frame content as the two other approaches are less reliable [96] or require specialized hardware [76].

In the context of wireless traffic, unlinkability [162] of frames implies that the attacker cannot distinguish whether they are related or not. This indistinguishability can be expressed as follows:

$$P(f_1 \sim f_2) = P(f_1 \not\sim f_2) = 1/2 \tag{III.2}$$

where $f_1 \sim f_2$ means that f_1 and f_2 are related and $f_1 \not\sim f_2$ signifies that they are not.

Let us consider that those frames are composed of n fields $\{h_i\}_{1 \leq i \leq n}$. Assuming that values of those in-frame fields are independent [1], unlinkability at the frame level and at the field level is equivalent:

$$P(f_1 \sim f_2) = P(f1 \not\sim f2) \Leftrightarrow \bigwedge_{1 \leq i \leq n} P(f_1.h_i \sim f_2.h_i) = P(f_1.h_i \not\sim f_2.h_i) \tag{III.3}$$

To enforce unlinkability of frames, it is thus sufficient to ensure that fields are unlinkable. In other words, if for each field h_i, the value of $f_1.h_i$ is unlinkable with $f_2.h_i$ then f_1 and f_2 are unlinkable.

III.5.2.2 Empirical unlinkability properties

Actually, the aforementioned properties can be used as a design help, but are not suitable for an empirical verification that would be performed on a sequence of frames. Indeed, the evaluation of the probabilities will be limited by practical constraints such as the duration of the observation and the frequency of identifier rotation. Therefore, we derived properties that can be applied to a sequence of limited size.

Let us consider a device d generating a sequence of frames f_i, each frame including a link layer identifier $f_i.addr$, as well as a set of n fields $\{f_i.h_j\}_{1 \leq j \leq n}$.

For any two consecutive frames f_{i_1} and f_{i_2} for which $f_{i_1}.addr \neq f_{i_2}.addr$ (link layer identifier rotation), the fields $\{h_j\}_{1 \leq j \leq n}$ of f_{i_1} and f_{i_2} must satisfy the following:

1. if h_j is an identifier or a data field: $f_{i_1}.h_j \neq f_{i_2}.h_j$
2. if h_j is a counter field modulo m: $d_m(f_{i_1}.h_j, f_{i_2}.h_j) > \delta$

---

1. This is usually the case in BLE and Wi-Fi discovery traffic.
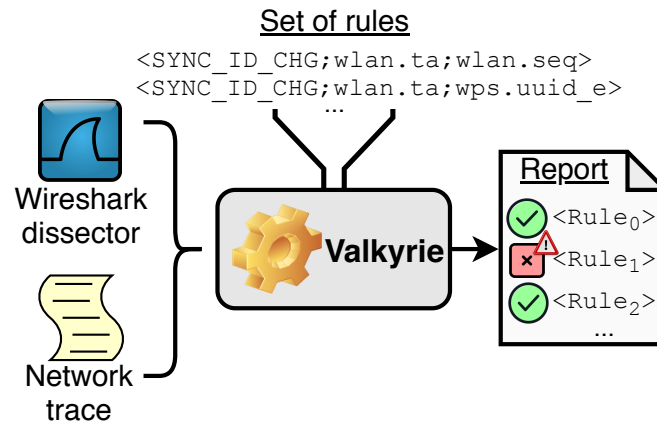
Figure III.15 – Functional diagram of *Valkyrie*. A network trace along with a set of rules are provided as inputs to the tool. The `Wireshark` dissector is leveraged for the protocol field denomination that must be specified in rules. At the end of the analysis, *Valkyrie* outputs a report specifying verified rules and breached ones with detailed warning messages.

where $d_m(x, y) = x - y$ if $x > y$ and $x + m - y$ otherwise, measures the distance between two values modulo m.

In the following, we will employ those empirical properties to identify issues in the address randomization implementations.

III.5.3   Design and implementation

In this section, we present the design of *Valkyrie* (*Verification of Addresses LinKabilitY in address Randomization ImplemEntations*), a software tool that can verify the enforcement of privacy properties on (wireless) network traffic traces.

As inputs, *Valkyrie* takes a network traffic trace generated by a device as well as a set of rules to be checked. Then, it verifies those rules independently and produces a set of warning messages for each breached rule (see Figure III.15). The code is available online [21].

III.5.3.1   Rules syntax

To specify rules presented in Section III.5.2, we designed a custom syntax. A rule specifies the link layer field that is rotating and upon which a property must be enforced: this field is called `address` in our syntax. Then, the rule needs to specify the field that must satisfy the property: this field is called the `target`. Each rule is also associated with a type, noted `type`, which defines the type of property that needs to be satisfied.

Currently, our tool includes two rule types: `SYNC_ID_CHG` and `SYNC_CNT_CHG`, which respectively cover the identifier/data and counter properties. Optional parameters can be appended to those rules: for instance, the distance δ in the case of the `SYNC_CNT_CHG` rule. Finally, a rule has the following form:

$$\texttt{type, address, target} <, \texttt{optional parameters} > \tag{III.4}$$

Valkyrie leverages on `pyshark` [163], a python wrapper for `tshark` (the command line version of `Wireshark` [164]), for the naming of frames and fields. As such, this means that the protocol field denomination used in the rules corresponds to the `Wireshark` one.

As a result, the tool can be applied to any of the `Wireshark` supported protocols, and even more by using *dissectors* which are frame parsers that can be written for any protocol. In this study, we wrote our custom dissector [22] to parse BLE messages of *Apple* Handoff, *Apple* Nearby Info and *Microsoft* CDP protocols (see Section III.5.4.3).

Syntax III.4 is then used to translate formal rules defined in Section III.5.2 into practical ones. As an example, in the case of 802.11, the counter rule applied to the `Sequence Number` can be written as:

$$< \text{SYNC\_CNT\_CHG}; \texttt{wlan.ta}; \texttt{wlan.seq} > \tag{III.5}$$

where `wlan.ta` designates the transmitter address of the device and `wlan.seq`, the `Sequence Number`.

III.5.3.2   Verification process

Given a network trace along with a set of rules, *Valkyrie* verifies that those rules are satisfied. Algorithm 1 describes this process which, for each rule, is performed as follows: for each consecutive frames f_1 and f_2 having distinct `address`, verify that $f_1.\texttt{target} \neq f_2.\texttt{target}$ in the case of `SYNC_ID_CHG`, and $d(f_1.\texttt{target}, f_2.\texttt{target}) > \delta$ in the case of `SYNC_CNT_CHG`.

To compute the distance between two values of a counter field, we consider that the counter is *looping*, i.e. it will go back down to zero after having reached the maximum value. Thus, the distance can be computed as presented in Section III.5.2.2.

III.5.3.3   Address reuse detection

In addition to those properties on the frame fields, *Valkyrie* also verifies that device addresses are not reused. More specifically, once used during a time interval, an address

Input: - Set of n rules $R = \{r_i\}_{0 \leq i < n}$
         - Network trace T composed of frames f_i
Output: Boolean vector V whose element $V[i]$ describes the satisfaction of rule r_i
foreach $r_i \in R$ **do**
     $V[i] = false$;
     foreach f_1 **and** $f_2 \in T$ **do**
         if f_1.address $!= f_2$.address **then**
             if (r_i.type == SYNC_ID_CHG **and** f_1.target $!= f_2$.target) **or**
             (r_i.type == SYNC_CNT_CHG **and** $d(f_1$.target$, f_2$.target$) > \delta$) **then**
                 $V[i] = true$;
             end
         end
     end
end

Algorithm 1: Verification algorithm of *Valkyrie*.

should not be reused later in order not to lead a passive eavesdropper to trivially link distinct frames broadcasted by the device. To this purpose, we provided *Valkyrie* with a feature that is able to detect address reuse by recording addresses appearing within a trace.

III.5.4 Experimental evaluation

In this section, we perform the evaluation of *Valkyrie* based on wireless traffic generated by real-world devices. To this end, we focus on two prominent IoT supported wireless technologies implementing address randomization: BLE and Wi-Fi.

III.5.4.1 Tested devices

Equipped with a BLE and/or a Wi-Fi interface, the evaluation is based on a set of 60 devices that can be categorized into three types: laptop, smartwatch and smartphone. This set covers major manufacturers such as *Apple*, *Google*, *Huawei*, *Motorola*, *Sony* and *Xiaomi*.

Some smartphones are tested with different OS versions. For instance, the *Apple* iPhone XR has been evaluated with iOS versions 12.1.2 and 12.4.1, while Android 7.1 and 9 have been experimented with the *Google* Pixel XL.

Table III.9 details the full list of tested devices that constitutes a representative sample of devices used in the world. Note that, all those devices are owned by the Furious MAC research group at the United States Naval Academy [165] or its institutions. To be

Table III.8 – List of specified rules for the experimental evaluation.

Benched element	Rule (🔴 : applied to Wi-Fi / 🔵 : applied to BLE)
WPS UUID (Wi-Fi)	① `<SYNC_ID_CHG;wlan.ta;wps.uuid_e>`
Auth Tag (*Apple* Nearby Info)	② `<SYNC_ID_CHG;bthci_evt.bd_addr;apple_nearby_info.auth_tag>`
Device Hash (*Microsoft* CDP)	③ `<SYNC_ID_CHG;bthci_evt.bd_addr;microsoft_cdp.device_hash>`
Sequence Number (Wi-Fi)	④ `<SYNC_CNT_CHG;wlan.ta;wlan.seq>`
IV (*Apple* Handoff)	⑤ `<SYNC_CNT_CHG;bthci_evt.bd_addr;apple_handoff.iv>`

transparent, this signifies that the corpus of traffic traces over which we tested *Valkyrie* was constituted by this research group prior to be graciously shared with us.

III.5.4.2   Traffic capture protocol

For each device, a traffic capture was obtained by isolating the device in a Faraday cage [1] and was then stored in the `pcap` format. This rules out the possibility that devices were connected to another device or an access point. As a consequence, they only generated discovery traffic: *advertisement packets* for BLE and *probe requests* for Wi-Fi. Moreover, during captures, devices were left untouched with their wireless interface (i.e. BLE or Wi-Fi) enabled. Note that, each capture lasts 20 minutes or gathers 200 frames, whichever is first.

III.5.4.3   Rules specifications

The verification process is based on a set of rules. Leveraging the language designed in Section III.5.3.1, Table III.8 specifies five rules corresponding to the five main issues affecting address randomization according to the literature (see Section III.5.1).

The three first rules cover issues related to *identifiers* in the frame body such as the WPS UUID field in Wi-Fi (①), and the Auth Tag and Device Hash respectively found in *Apple* Nearby Info (②) and *Microsoft* CDP (③) BLE messages. The last two rules cover predictable *counter* fields, namely the Sequence Number in Wi-Fi (④) and the IV in *Apple* Handoff BLE messages (⑤).

---

1. A Faraday cage is an enclosure used to block the entry or escape of an electromagnetic field. In our study, it has been leveraged to prevent external radio signals from being added to the traffic captured from the benched device.

III.5.4.4 Results

For each capture, we ran *Valkyrie* loaded with the rules set corresponding to the wireless technology: rules ① and ④ for Wi-Fi, and ②, ③ and ⑤ for BLE. For each rule, Table III.9 gathers raised issues. Note that, an issue is raised if the rule is unsatisfied at least once.

A first observation is that all devices are affected by at least one issue, and that more than 73% are affected by two or more.

In Wi-Fi, the most prevalent issue is the non-reset `Sequence Number` (④), which affects 98.3% of devices. The results on smartphones experimented with different versions of their OS such as *Apple* iPhone 5S and *Google* Pixel XL show that software updates hampered tracking based on this `Sequence Number`.

However, although this issue was supposed to be corrected in version 8 of Android [166], some devices running this OS version such as *Huawei* P20 Lite and *Sony* Xperia XZ1 are still affected. In [81], Martin et al. already identified this address randomization misimplementation that seems to be related to the manufacturer. Finally, 8.3% of tested devices are prone to the static `WPS UUID` issue (①).

In BLE, all *Apple* devices except the *Apple* MacBook Pro laptop match with corresponding rules ② and ⑤. In fact, the *Apple* MacBook Pro is not affected by rule ② as it does not contain any `Auth Tag` in its emitted *Apple* Nearby Info BLE messages.

Similarly, rule ③ is only raising an issue with the *Dell* G3 laptop broadcasting *Microsoft* CDP frames, which is the only device running Windows. As a result, *Valkyrie* verified the expected non-reset counter and static identifier concerns in which all *Apple* and *Microsoft* benched BLE devices expose their owners to tracking.

Lastly, *Valkyrie* detected that 45% of devices reuse random device addresses, especially smartphones of manufacturers *Apple*, *ASUS*, *Blackberry*, *HTC*, *Huawei*, *LG*, *Motorola*, *Sony*, *Xiaomi* and *ZTE* (see Table III.9). De facto, it is unclear why a device reuses an address. The possible explanations include: poor PRNGs used for address generation or a switch to a static address of the device.

Note that, given the limited length (i.e. 20 minutes) of the capture, results may include false negatives: some devices might be breaking one of the rules, but the capture was not long enough to catch this behavior. For instance, the `Sequence Number` issue (④) was not found in the capture from the *Apple* iPhone 5S running iOS 11.2.1, while it appears not to have been fixed until iOS 13.1 at least.

Table III.9 – List of evaluated devices along with their OS versions and identified issues. Gray lines depict a software update of the involved devices.

Type	Device	OS version	Identifier ①	②	③	Counter ④	⑤	Addr. reuse
Laptop	Apple MacBook Pro (13", 2015)	macOS 10.13.6				✓	✓	
	Dell G3 17-3779	Win 10 Pro (v1809)	✓		✓	✓		
	HP EliteBook Folio 1040 G3	Ubuntu 16.04.6 LTS				✓		
Watch	Apple Watch Series 2	watchOS 5.0.1		✓		✓	✓	
	Apple Watch Series 3	watchOS 5.1.3		✓		✓	✓	
Phone	Apple iPhone 5C	iOS 9.3.1		✓		✓	✓	
	Apple iPhone 5S	iOS 10.3.2		✓		✓	✓	
	Apple iPhone 5S	iOS 11.2.1		✓			✓	
	Apple iPhone 5SE	iOS 11.3		✓		✓	✓	
	Apple iPhone 6	iOS 12.1		✓		✓	✓	✓
	Apple iPhone 6S	iOS 11.4		✓		✓	✓	
	Apple iPhone 6S Plus	iOS 12		✓		✓	✓	
	Apple iPhone 7	iOS 11.2.6		✓		✓	✓	
	Apple iPhone 7 Plus	iOS 12.0.1		✓		✓	✓	
	Apple iPhone 8 Plus	iOS 11.4.1		✓		✓	✓	
	Apple iPhone XR	iOS 12.1.2		✓		✓	✓	
	Apple iPhone XR	iOS 12.4.1		✓			✓	
	Apple iPhone XS	iOS 13.1		✓		✓	✓	
	Apple iPhone XS Max	iOS 12.1		✓		✓	✓	
	Aquos sense	Android 8.0.0				✓		
	ASUS Zenfone 3	Android 7				✓		✓
	ASUS Zenfone 3 Deluxe	Android 6.0.1				✓		✓
	Blackberry Privilege	Android 5.1.1	✓			✓		✓
	Google Pixel XL	Android 7.1				✓		
	Google Pixel XL	Android 9						
	HTC One A9	Android 6				✓		✓
	HTC U11	Android 7.1.1				✓		
	Huawei Mate10 lite	Android 7				✓		
	Huawei Nexus 6P	Android 6.0.1	✓			✓		✓
	Huawei P10 Lite	Android 7				✓		
	Huawei P20 Lite	Android 8.0.0				✓		✓
	Huawei P9	Android 6				✓		✓
	Huawei P9 Lite	Android 6	✓			✓		
	Huawei Y7 Prime (2018)	Android 8.0.0				✓		
	LG V20	Android 7				✓		✓
	Motorola Moto G Play (6th gen.)	Android 8.0.0				✓		✓
	Motorola Moto e	Android 5.1				✓		✓
	Motorola Moto E (4th gen.)	Android 7.1.1				✓		
	Motorola Moto E Plus (4th gen.)	Android 7.1.1				✓		✓
	Motorola Moto G (3rd gen.)	Android 5.1.1				✓		✓
	Motorola Moto G (4th gen.) Plus	Android 6.0.1				✓		
	Motorola Moto G (5th gen.)	Android 7				✓		✓
	Motorola Moto G (5th gen.) Plus	Android 7				✓		✓
	Motorola Moto G4 Plus	Android 6.0.1				✓		✓
	Motorola Moto G5	Android 7				✓		
	Motorola Moto G5 Plus	Android 7				✓		✓
	Motorola Moto GS (5th gen.)	Android 7.1.1				✓		✓
	Motorola Moto Z Play	Android 6.0.1				✓		
	Motorola Moto Z Play	Android 9						
	Motorola Nexus 6	Android 7	✓					✓
	OnePlus 2	Android 6.0.1				✓		
	OnePlus 3T	Android 7				✓		
	Sony Xperia X Compact	Android 7				✓		
	Sony Xperia X Compact	Android 8.0.0						
	Sony Xperia XZ Premium	Android 8.0.0				✓		✓
	Sony Xperia XZ1	Android 8.0.0				✓		
	Xiaomi Mi 5	Android 7				✓		
	Xiaomi Mi A1	Android 7.1.2				✓		
	Xiaomi Mi A1	Android 8.0.0						
	Xiaomi Redmi 3S	Android 6.0.1				✓		✓
	Xiaomi Redmi 4A	Android 7.1.2				✓		✓
	Xiaomi Redmi 4X	Android 7.1.2				✓		✓
	Xiaomi Redmi 5 Plus	Android 7.1.2				✓		✓
	Xiaomi Redmi 5A	Android 7.1.2				✓		✓
	ZTE Blade X Max	Android 7.1.1				✓		✓
	ZTE Grand X 4	Android 6.0.1				✓		✓

III.5.4.5   Evaluation summary

To put in a nutshell, the evaluation demonstrates that the current implementation of *Valkyrie* is usable and allows to detect most privacy-threatening behaviors such as static identifiers and non-reset counters. Furthermore, the proposed rule specification language was flexible enough to express associated requirements.

III.6   Conclusion

To conclude, this chapter aimed to demonstrate how *passive* and *active* attackers can track users leveraging advertising data and metadata broadcasted by their carried BLE devices. Overall, the detailed contributions complement previous works [77, 81, 97, 1] that revealed the difficulties in the implementation of the device address randomization.

First, we analyzed the privacy provisions and issues in current implementations of the BLE advertising mechanism. In particular, even if address randomization is widely adopted by vendors, we discovered that a number of devices still use Stable addresses, thus exposing their owners to tracking. Furthermore, we found that some devices exceed the recommended maximum duration of the random addresses.

Despite the use of a random device address, we also identified that the content of the advertisement packets can be used to track users. Indeed, the address randomization scheme can be rendered useless by static identifiers and non-reset counters included by some devices in their advertising data. Especially, we showed that custom data embedded in proximity protocols of *Apple*, *Microsoft* and *Google* can be leveraged to defeat the address randomization.

Note that, on April 5th, 2019, we informed the manufacturers[1] of the identified privacy issues. However, as of July 7th, 2020, none[2] of those issues have been addressed.

Actually, a part of the presented issues are the results of manufacturers that do not follow the Bluetooth Core Specification. For instance, the inclusion of the `LE Bluetooth Device Address` AD structure in advertising data is against the specifications.

Nevertheless, most issues are the results of practices that are not forbidden by the specifications (e.g. using a custom UUID that includes a device address is compliant with the specifications). Similarly, there are no instructions about the nature of the information that can be carried by the `Manufacturer Specific Data` and `Service Data-16-bit UUID` AD structures. With regard to the privacy implications, the Bluetooth Core Specification does not provide any guidelines about the content of the advertising payload.

---

1. Notifications have been sent to: *Apple*, *Arcadyan*, *Boosted Boards*, *Google*, *LG Innotek*, *Microsoft*, *Nokia/Withings*, *Samsung*, *Sony* and *Xiaomi*.
2. As far as we know.

Second, for the sake of transparency, we considered that it is necessary to show the general public how physical tracking can stem from wireless technologies. As a consequence, we introduced *Venom*, a *Visual and ExperimeNtal BluetOoth Low Energy tracking systeM* that aims to 1) raise user awareness about radio based physical tracking technologies and 2) experiment privacy-preserving mechanisms. As opposed to *Wombat* [154], its counterpart applied to the Wi-Fi technology, *Venom* leverages a wireless infrastructure to track users through their BLE enabled devices.

To outline the fundamentals, we described the architecture of this system before to present how it can be leveraged for demonstrational purposes. Thereafter, we explained how *Venom* can be used to deploy and test privacy-enhancing features. To this end, we described a primitive and easy-to-use BLE based opt-out mechanism allowing users to express their dissent to the tracking.

Beyond public awareness, the objective of *Venom* is to support that it is imperative to complement the Bluetooth Core Specification with additional requirements that will cover privacy issues on the BLE protocol, and especially on the advertising mechanism.

Third, we studied data exposed within GATT profiles of BLE devices. Particularly, we demonstrated that the content of a GATT profile can be leveraged to fingerprint a device. In fact, identifiers and values composing this profile are diverse enough to act as a fingerprint. Even if it features anti-tracking mechanisms such as the device address randomization, this fingerprint can be then used to track a device.

To the best of our knowledge, there is no indication that those fingerprinting techniques are currently exploited in the wild. In the web ecosystem, introduction of anti-tracking techniques has triggered [167] the deployment of fingerprinting techniques by web trackers. The recent adoption of address randomization in wireless technologies [77, 1] can trigger a similar move by the industry of physical tracking. Once more, taking this threat into account by reviewing and complementing the Bluetooth Core Specification with additional requirements is needed.

Lastly, we presented the first attempt at automatically verifying the correctness of address randomization implementation. To this purpose, we discussed requirements for protecting users against tracking, and derived a list of properties leveraging works done by the community.

Afterwards, we prototyped *Valkyrie* (*Verification of Addresses LinKabilitY in address Randomization ImplemEntations*), a versatile tool able to verify properties written in a `Wireshark` based language. Based on the previously identified issues within address randomization implementation (i.e. static identifiers and non-reset counters), we showed that properties associated with such issues can be expressed using this language. Relying on a representative set of BLE and Wi-Fi enabled devices, we evaluated the proposed tool demonstrating that *Valkyrie* was able to detect issues in the generated wireless traffic.

As such, the developed approach can be applied by vendors to verify that privacy properties

are enforced by their devices. In addition, it can be included as a part of a certification process to verify that some devices are meeting privacy requirements. Finally, provided that it is supported by `Wireshark` or that a dissector exists, we point out that this approach can be adapted to any protocol.

Chapter IV

Inferring users information from Bluetooth/BLE wireless communications

This chapter demonstrates how passive and active eavesdroppers can exploit wireless communications of Bluetooth/BLE enabled devices to infer information on their corresponding users. To begin, the adoption of the Bluetooth/BLE technology along with its privacy implications are introduced in Section IV.1. Thereafter, Section IV.2 details the threat model that is considered throughout this chapter. Users sensitive information such as personal characteristics and medical conditions that can leak from passive observations of the BLE advertisement traffic are described in Section IV.3. Section IV.4 shows that data exposed within GATT profiles can be actively mined to infer a plethora of users information too, while the activity inference through a Bluetooth based timing attack is analyzed in Section IV.4.3. Section IV.5 presents Himiko, a Human Interface for Monitoring and Inferring Knowledge on Bluetooth Low Energy Objects. With regard to the reported privacy vulnerabilities, a set of recommendations that could be addressed by manufacturers as well as the Bluetooth SIG is provided in Section IV.6. To finish, Section IV.7 concludes this chapter.

Corresponding contributions : [8, 7, 6, 9]

IV.1 Introduction

Bluetooth is a 2.4 GHz ISM radio communication standard developed by the Bluetooth SIG to exchange data over a short range [1]. Jointly released with the Bluetooth version

---

1. From one meter to a hundred meters, depending on the transmitter characteristics [38, Vol 2, Part A, sec. 3].

4.0 [39], BLE brings several improvements including a lower power consumption at the cost of a reduced bandwidth.

As a result, BLE has been adopted for most battery-powered devices such as smartphones, smartwatches, fitness wristbands, headphones, etc. According to the Bluetooth SIG, more than four billion devices supporting Bluetooth/BLE have been shipped in 2019 [168]. Therefore, Bluetooth is becoming a key element in daily lives where users are likely to leave their Bluetooth interfaces enabled most of the time.

Because of its massive adoption, Bluetooth has been subject to various attacks aiming to compromise security and, more particularly, privacy of users. For instance, Bluetooth has been exploited to track location of users [121, 85, 123], infer their physical activities [143] and even profile them through *inventory attacks* [59].

To prevent those threats, privacy-preserving features have been integrated into the Bluetooth Core Specification. LE Privacy [38, Vol 3, Part C, sec. 10.7], the most important feature, defines the use of temporary link layer identifiers that substitute the device address with random temporary pseudonyms. However, despite those improvements, it has been shown [143, 59] that some information can still leak from Bluetooth/BLE devices.

In this chapter, we disclose privacy leaks that stem from *activity inference* and *inventory attacks*. To this purpose, we rely on the two datasets – $dataset_{Passive}$ and $dataset_{Active}$ – previously used in Chapter III. Constituted of real-world observations of advertisement packets and GATT profiles collected over a period of five months, we remind that those datasets are respectively associated with more than 53500 and 13200 different device addresses (see Table III.2).

First, we introduce our threat model that considers the case of Bluetooth/BLE devices announcing themselves by broadcasting advertisement packets, and including privacy issues in their exposed data. To pursue, we assume an attacker that can be both *passive* (i.e. he only scans for observable data, hiding its presence on the wireless channel) and *active* (i.e. he can exchange Bluetooth frames with its target). Moreover, we limit the presented attacks solely to the inference of users information. As users profiling results from the gathering then correlation of multiple inferred information, this process is thus not considered.

Second, as collection of information on users in the physical world is a growing trend, companies can leverage the opportunities offered by wireless devices to track and profile users. Leveraging $dataset_{Passive}$, we support this statement identifying several elements that can be used to infer information about the owner of a BLE enabled device. In particular, we show that advertisement packets include data that can reveal the manufacturer, model and type of the device as well as names of users, exposing them to *inventory attacks* and inference of sensitive attributes. For instance, some medical devices are broadcasting their types (e.g. hearing aid, glucometer, insulin pen, etc.) trivially betraying a medical condition of their corresponding owners.

Third, leveraging $dataset_{Active}$, we begin by highlighting that data included in GATT profiles can be mostly read without authentication. As a reminder, a GATT profile is a structure containing services and characteristics supported by the BLE device, and that has the possibility to deny reading of data to unauthenticated devices leveraging access control properties (see Section I.2.2.4). From this observation, we shed light on several issues where such profiles can be mined to infer sensitive information such as personal characteristics impacting privacy of users. Note that, we draw the attention to the fact that a part of those information can be inferred from advertisement packets found within $dataset_{Passive}$ too. Indeed, the content of the advertisement packets can be assimilated as a brief summary of the information exposed in corresponding GATT profiles. As such, this means that advertisement packets and GATT profiles share elements such as device names and service UUIDs on which we based our observations to infer users information.

During this study, we also discover that manufacturers such as LG can divert the use of Bluetooth SIG defined characteristics for other purposes, raising serious privacy issues at the same time. Overall, results of this work point out that provisions of the LE Privacy alone are not enough to guarantee privacy of users.

Fourth, we consider a scenario in which the attacker is interested in learning information on the *activity* of a remote Bluetooth enabled device. In practice, a part of the Bluetooth operations are handled by the operating system and are relying on resources shared with the rest of the system. The speed at which the Bluetooth tasks are handled is thus affected by the overall load of the system. Based on this idea, we demonstrate an *active attack* that allows a remote attacker to infer information on state changes by leveraging the Bluetooth connectivity of the targeted device. To this purpose, this attack relies on a request-response mechanism (ping) provided by the L2CAP layer of the Bluetooth. By flooding the target with requests while recording the round-trip time (RTT), the attacker can analyze the timing information to detect device state changes. To gather additional information about the user, those state changes could be further correlated with other sources of information (e.g. visual observations).

Last but not least, to both raise awareness about the privacy issues that the BLE advertising mechanism can involve and alert on the need to complement privacy considerations of the Bluetooth Core Specification, we implement *Himiko* (*Human Interface for Monitoring and Inferring Knowledge on Bluetooth Low Energy Objects*). In fact, this tool aims to expose the information that an eavesdropper can infer by leveraging the content of advertisement packets and GATT profiles of BLE enabled devices. The advertising raw data are collected and processed from devices that have their Bluetooth interface enabled. Afterwards, the user is shown the information that are leaking from his device.

Finally, we suggest a set of solutions to mitigate reported privacy vulnerabilities. In this regard, we discuss countermeasures that could be implemented by manufacturers, and focus on improvements of the Bluetooth Core Specification to strengthen discovery procedures of GATT profiles.

IV.2 Threat model

A wide variety of sensors can be found in consumer IoT devices. Depending on the type of device and its field of application, emitted data can be exploited by eavesdroppers to infer information about activities (e.g. running, sleeping, cooking, etc.) but also attributes (e.g. deafness, heart disease, music lover, etc.) of individuals. As a consequence, wireless communications from IoT devices can raise privacy concerns.

In particular, this is the case of inference attacks such as *activity inference* and *inventory attack* (see Section II.4) that can leverage data mining techniques to analyze data broadcasted by IoT devices and infer personal information on their corresponding owners. Actually, the main objective of such inference attacks is to piece together technical information such as the type and manufacturer of a device in order to determine a more sensitive information. For instance, a connected glucometer can betray a diabetic user, while someone carrying a *Nintendo* device is likely a gamer.

In this chapter, we study wireless communications from Bluetooth/BLE enabled devices, and especially privacy issues included in their advertising data and patterns.

To this end, we consider an external attacker which continuously monitors the advertising traffic on Bluetooth/BLE channels. Depending on the triggered attack, we assume that this attacker can be either *passive* or *active*. In some cases, this means that interactions with remote devices are considered (see Section IV.4). However, we point out that the attacker never interacts with the wireless channel through injection or jamming.

As such, the attacker is then able to 1) capture the advertisement packets broadcasted by nearby BLE enabled devices and 2) exchange Bluetooth frames with its target. Note that, those assumptions can be satisfied using off-the-shelf hardware and open-source software. As described in Section III.2.1, a 10$ CSR v4.0 Bluetooth USB dongle along with Bluetooth tools featured GNU/Linux distribution is enough to meet those requirements.

Concerning the content of the BLE wireless communications, advertising payloads are assumed to be in clear, encrypted or both [1]. Therefore, the attacker can sometimes have access to cleartext information such as a blood sugar level and pressure respectively broadcasted from a glucometer and a blood pressure sensor, for instance.

Nevertheless, we assume that the device used by the attacker has not been previously paired with any Peripheral: it cannot authenticate itself and access protected values of characteristics. Note that, available in clear, packet header information and traffic metadata such as speed rate and periodicity of data emissions are also part of our studies.

On the victim side, we only make the assumptions that his device has its Bluetooth interface turned on and is in communication range. It is important to mention that the

---

[1]. Some elements embedded within the advertising payload are in clear while others are encrypted (see Section III.3.4.2.1 and Section III.3.4.2.2).

short range of commodity Bluetooth interfaces can be a limitation to the impact of the attacks exposed in this chapter. Nonetheless, this range can be significantly extended [1] with the help of custom hardware solutions [169, 170].

Definitely, the goal of our modeled attacker is to identify and classify signals of Bluetooth/BLE enabled devices. More specifically, leveraging observations on the advertising data and metadata, a successful attack is defined as the attacker being able to infer sensitive information that can disclose ongoing activities and/or user related characteristics such as his interests, his character traits and his medical condition.

Finally, attacks described in this chapter can constitute a first step toward more elaborated attacks. Indeed, combined with another source of information, they can be leveraged to further identify a user. As an example, the detected state changes of a device can be combined with visual observations of a set of smartphone users [68]. The attacker can then correlate observable actions (e.g. smartphone in pocket, smartphone in use, etc.) to establish a link between a physical entity (i.e. a user) and a network identifier (i.e. a BD_ADDR).

Note that, being able to identify and recognize users through their broadcasted data can constitute a basis for profiling [101]. However, user profiling is an advanced privacy threat that we will not deal with in this thesis. In fact, only users information that can be directly inferred from the observations of advertising data and metadata are studied.

IV.3   Passive information inference based on advertising data

In this section, we study privacy threats stemming from the BLE advertising mechanism, leveraging $dataset_{Passive}$ (see Section III.2). To this purpose, we demonstrate how some elements of advertising data can be exploited to infer information on manufacturers prior to show how some others can be mined to identify the type of device, exposing the owner to *inventory attacks*.

IV.3.1   Inferring information on manufacturers

In this section, we detail how data carried by advertisement packets found within $dataset_{Passive}$ can leak information on manufacturers. As a reminder, $dataset_{Passive}$ has been collected over five months, contains about eight million records and includes more than 53500 distinct BD_ADDR (see Section III.2). Moreover, since some of those BD_ADDR are random pseudonyms, the number of actual devices is expected to be smaller than the number of distinct device addresses.

---

1. From a hundred meters to a mile.
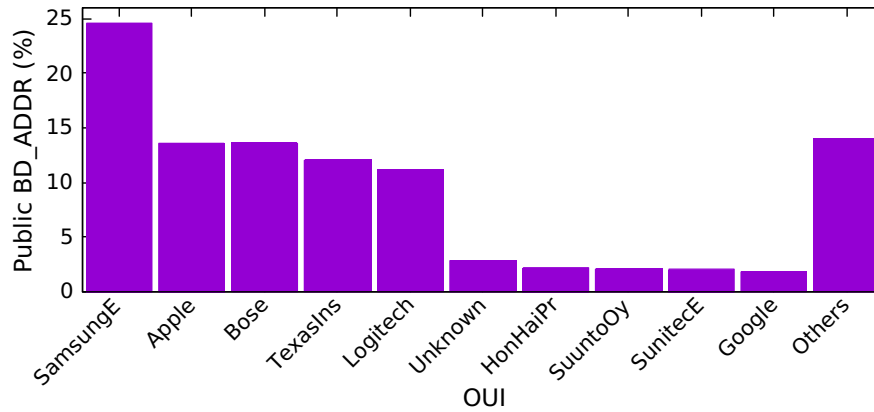
Figure IV.1 − Distribution of OUI among Public device addresses.

IV.3.1.1   Manufacturers OUI

By definition, for device addresses falling in the Public type, it is possible to identify the manufacturer via the OUI of the address. As a result, Figure IV.1 presents the distribution of the top 10 OUI found within Public addresses.

First, we can observe the prevalence of *Samsung* and *Apple*, two companies producing electronic devices such as smartphones and tablets. *Bose*, a known company for its high quality headsets, comes third with 13.6% of Public addresses of $dataset_{Passive}$. Then, OUI such as `TexasIns` and `SuuntoOy` respectively betray manufacturers of chipsets and fitness trackers.

Also, we note the presence of *Google* with 1.82% of Public addresses. Leveraging device names broadcasted by such items, we found they are mostly *Google* Chromecast (see Section III.3.4.2.3). In addition, even though they are not made to be carried by users over time, we show that they can be leveraged to determine identities of their owners as they can be customized to embed first names and last names in their advertising data (see Section IV.3.2.3).

As a side note, because they configure their products to use Public addresses, we point out the fact that all those companies are trivially exposing their customers to physical tracking.

IV.3.1.2   Company Identifiers

Company Identifiers (IDs) are 16-bit numbers assigned by the Bluetooth SIG to companies [171]. Actually, such identifiers are included in the `Manufacturer Specific Data` AD structure that is dedicated to carry custom data of manufacturers.

Table IV.1 – Distribution of company IDs assigned by the Bluetooth SIG among registered OUI of Public addresses. Numerical values are numbers of Public addresses.

Co. ID / OUI	Samsung	Apple	IBM	SGLItalia	ARTiming	Harman	Sony	Google	TomTom	Suunto
SamsungE	1022									
Apple		600								
Logitech			341							
Bose				261						
TexasIns					187				2	
SunitecE						104				
HonHaiPr		40					47	5		
Google								44		
TomtomSo									41	
SuuntoOy										36

As such, the presence of a company ID suggests that the device is manufactured by this company. However, it may not always be the case as third party manufacturers may use company IDs to enable compatibility with the feature.

As an example, leveraging our laboratory hardware, we discovered that *Chipolo* keyrings advertise the *Apple* company ID (`0x004c`) followed by the format of the *Apple* iBeacon `Manufacturer Specific Data` AD structure for localization purposes. In $dataset_{Passive}$, we also found that *Fossil* Q Venture smartwatches broadcast data related to the *Google* company ID (`0x00e0`) suggesting that *Fossil* uses a protocol of *Google* to be compatible with Android smartphones.

To push the study further, we computed the distribution of company IDs assigned by the Bluetooth SIG among registered OUI of Public addresses. From our experimental results (see Table IV.1), we can observe that manufacturers such as *Logitech* and *Bose* do not seem to include the company IDs corresponding to their OUI, unlike *Samsung* and *Apple*, the two predominant manufacturers in $dataset_{Passive}$. Thus, for a number of companies, the company ID found within advertising data can be an indicator of the manufacturer.

In addition to those observations, we considered the distribution of those company IDs among the device address types. Figure IV.2 shows the distribution of Stable and Private device addresses that match a company ID assigned by the Bluetooth SIG among all Stable and Private BD_ADDR.

At first, we can see that some company IDs are associated with one category of addresses. For instance, the *Samsung* company ID is only associated with Stable addresses and is seldom seen with Private addresses. The opposite trend can be observed for the *Microsoft* company ID that is only associated with Private addresses. For other companies such as *Apple*, both Stable and Private addresses are associated with the company ID suggesting a diversity in the implementations.

(a) Stable addresses (4300 BD_ADDR)   **(b)** Private addresses (45300 BD_ADDR)
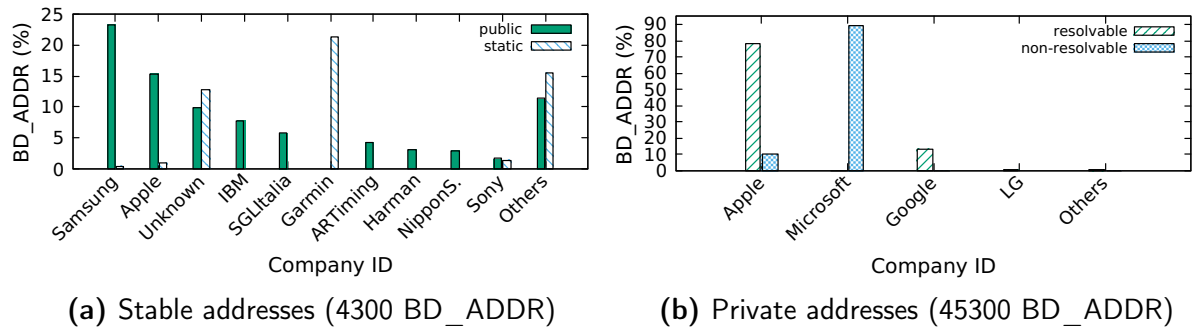
Figure IV.2 – Distribution of the Bluetooth SIG assigned company IDs among Stable and Private addresses.

Table IV.2 – Summary of the information that can be inferred from BLE advertising data.

Advertising element	Inferred information			
	Manufacturer	Model	Type	Owner identity
Public BD_ADDR	✓	✓	✓	
Device names	✓	✓	✓	✓
Manufacturer Specific Data	✓	✓	✓	
Company IDs	✓		✓	
Manufacturer data	✓	✓	✓	
Service UUIDs	✓	✓	✓	
Appearance			✓	
Class of Device			✓	

Focusing on Private addresses (see Figure IV.2b), we can witness that three companies (i.e. *Apple*, *Microsoft* and *Google*) account for the majority of the records. Especially, *Apple* represents more than 52% (2.5M advertisement records).

IV.3.2   Inferring information on devices and their owners

BLE advertising data include information that can cause privacy threats beyond physical tracking. Indeed, we discovered that the characteristics of a device such as its manufacturer, model or type can be inferred. In the context of *inventory attacks* and profiling [82], those elements of information can become a privacy threat where an attacker will try to infer information based on advertising data broadcasted by the device carried by the user. For instance, a medical device can reveal a medical condition, and profiling individuals based on their devices is a direction taken by some companies [78].

In this section, we present a review of the information that can be inferred from advertising data. Our findings are summarized in Table IV.2.

IV.3.2.1   Device model and manufacturer

Model and manufacturer of BLE enabled devices are information that can be inferred from multiple advertising data elements.

Public BD_ADDR: The Public device address is an identifier that can be exposed in the `AdvA` field of the advertising payload (see Section I.2.2.3.2) but also in other AD structures (see Section III.3.4.1). In fact, such an identifier can be leveraged to determine the device manufacturer because, as a MAC address [38, Vol 2, Part B, sec. 1.2], its 24 MSB part corresponds to the OUI of the manufacturer.

Furthermore, since MAC addresses are typically allocated by batch, a Public address can be used to identify the device model. This approach has been demonstrated with Wi-Fi MAC addresses [172] and can be reproduced with BLE Public BD_ADDR. In addition, as shown in Section III.3.3.2, some Random Static addresses appear to be allocated by range as well. In those cases, the device model can be also inferred from the Random Static address.

Device names: As human friendly descriptors, the `Complete Local Name` and `Shortened Local Name` [146, Part A, sec. 1.2] AD structures often include identifiers of manufacturers and models of devices. As opposed to Public device addresses, there is no standardized nomenclature of the manufacturer. However, device names carried by those two AD structures are a rich source of information that can be correlated with online marketplaces such as *Google* Shopping [173] and *Amazon* [174] to deduce the manufacturer.

For instance, we identified that devices including patterns such as `LE-Bose` and `Jabra` in their device names are respectively *Bose* and *Jabra* manufactured headsets (see Table C.1 in Appendix C for an extended list of discovered patterns of device names). Moreover, we found that devices such as *Garmin* fitness trackers and *Bang&Olufsen* headphones include the name of their models in their device names.

Company IDs and service UUIDs: Advertising data can include several identifiers tight to companies. In particular, this is the case of company IDs that are included in the `Manufacturer Specific Data` AD structure. Like company IDs, members UUIDs are 16-bit identifiers published by the Bluetooth SIG [175]. Such members UUIDs can be included in the `Service Data-16-bit UUID` and `Complete List of 16-bit Service Class UUIDs` AD structures. In fact, the presence of those indicators suggests that the device features a service that has been designed by the corresponding company.

Furthermore, some companies can design their own 16-bit UUIDs. Although not publicly available, this information can be obtained by analyzing the advertisement traffic generated by the devices. Similarly, 128-bit service UUIDs can be generated by manufacturers and can be leveraged as well. For instance, the `adab09ad-6e7d-4601-bda2-bffaa68956ba` custom 128-bit service UUID broadcasted by *Fitbit* fitness trackers is associated with the *Fitbit* manufacturer (see Table C.2 and Table C.3 in Appendix C for respectively an

extended list of discovered custom 16-bit and 128-bit service UUIDs).

For the sake of compatibility, it is possible that a service is implemented by other companies than the one that has designed it. Nonetheless, in practice, we witness that those identifiers are generally included by devices of the associated manufacturer (see Section IV.3.1.2). Thus, company IDs and members UUIDs can be indicators of the company that has produced the device.

From $dataset_{Passive}$, we computed that more than 78% of Stable and 95% of Private addresses are associated with at least one company identifier (i.e. company ID or members UUID) in the advertising payload, indicating that the manufacturer can be identified.

`Manufacturer Specific Data:` In addition to the company ID, data embedded in the `Manufacturer Specific Data` AD structure can include fields indicating the model of the device. Indeed, this is the case of *Garmin* (company ID `0x0087`), for which the 16 MSB of data disclose the model of the device: `0x0657` and `0x0802` respectively indicate a *Garmin* Forerunner 620 and a *Garmin* Fenix 3 fitness smartwatch.

IV.3.2.2   Device type

As for the model and manufacturer, the type of a device is an information that can be determined mining the advertising data.

Public BD_ADDR: From previous observations, the manufacturer of a device can be derived from its Public BD_ADDR. Nevertheless, when this manufacturer is focused on a single class of products, it becomes possible to infer the type of the device from the Public address. For instance, OUI associated with *Nintendo* and *OculusVR* respectively reveal a video game platform and a virtual reality headset.

Company IDs: Carried by the `Manufacturer Specific Data` AD structure, company IDs can also constitute a source of information for the identification of the device type. Leveraging the core business of a manufacturer, we found that company IDs such as `0x02f2`, `0x0321` and `0x036d` respectively betray *GoPro* cameras, *Jewelbots* smart wristbands and *AirBolt* smartlocks.

Service UUIDs: Service UUIDs are processed by BLE enabled devices to discover services offered by nearby Peripherals before to establish a connection. In fact, some service UUIDs assigned by the Bluetooth SIG reveal particular characteristics of a device that can lead to the inference of its type. As an example, looking at the online Bluetooth SIG database of 16-bit service UUIDs [44], we can infer that devices which broadcast the `Running Speed and Cadence` service UUID (`0x1814`) are likely fitness trackers, while those that advertise the `Insulin Delivery` service UUID (`0x183a`) are healthcare devices.

Moreover, as mentioned in Section IV.3.2.1, a custom 128-bit service UUID can disclose

attributes of a device such as its manufacturer and its model that can lead to an implicit identification of its type. For instance, the `aa745be2-9025-4bf2-a318-91f3dba2999f` 128-bit service UUID is associated with *Garmin* Nuvi GPS, while `0000de00-3dd4-4255-8d62-6dc7b9bd5561` is advertised by *Nikon* cameras.

Nonetheless, even though those service UUIDs are customized by the manufacturers, we discovered that some of them are hardcoded in open-source specifications and codes. This is particularly the case with the *Google* hearing aid specifications [176] and the *Sony* SmartBand SWR-10 open API [177]. As a consequence, such online resources can be leveraged as well to identify the type of a device.

`Manufacturer Specific Data:` In addition to disclose the model of the device (see Section IV.3.2.1), the `Manufacturer Specific Data` AD structure can embed data that betray the type of the device. Actually, we found that the *Microsoft* CDP protocol includes a `Device Type` field (see Figure III.8) that provides information on the nature of the device. As a consequence, when the value of this field is filled accordingly to the *Microsoft* CDP specifications [148, sec. 2.2.2.2.3], we can infer the type of the advertising device by lookup in the specifications: Xbox One (`0x01`), Windows 10 Desktop (`0x09`), Windows 10 Phone (`0x0b`), etc. Analogously, we observed that *Apple* Continuity protocols can leak the device type (and model) through the `Manufacturer Specific Data` AD structure (see Section V.5).

`Appearance` **and** `Class of Device:` The Bluetooth Core Specification defines `Appearance` [38, Vol 3, Part C, sec. 12.2] and `Class of Device` [38, Vol 3, Part C, sec. 3.2.4] as two AD structures that carry information about the nature of a device, and more specifically its external appearance.

The level of information provided by values of those AD structures can be general (e.g. tag, clock, etc.) but also precise. For instance, the `Appearance` code `0x00c1` disclose a sport watch, while the `Class of Device` code `0x022808` indicates a toy vehicle. More worrisome, certain codes indicate specific medical devices: `Appearance` codes such as `0x0d00` and `0x0d48` respectively betray a glucose monitor and an insulin pen, while the `Class of Device` code `0x022930` denotes a knee prosthesis.

According to the Bluetooth Core Specification [146, Part A, sec. 1], the `Appearance` AD structure is not considered as sensitive and can be included in advertisement packets. On the contrary, no such specification can be found for the `Class of Device` AD structure, leaving the freedom of decision to manufacturers to include such a structure in their advertisement packets.

In $dataset_{Passive}$, we found that more than 5.3% of Public and 13.3% of Random Static addresses include either an `Appearance` or a `Class of Device` AD structure in their advertising data (see Table D.1 in Appendix D). In addition, for Private addresses, only 0.01% of Random Resolvable BD_ADDR broadcast data carried by the `Appearance` AD structure, while the `Class of Device` type is not advertised.

Among the most common appearances and class of devices, we found that devices such as *Tile* iTag keyrings, *Giant* RideSense cycling sensors and *Diggro* ID10x heart rate belts explicitly advertise their external appearances allowing a *passive* eavesdropper to infer the type of the device based on those information.

IV.3.2.3 Identity of the owner

The `Complete Local Name` and `Shortened Local Name` AD structures carry a textual description of the device that can be often customized by users. As a result, a number of those device names include the identity of the owner under the form of a full name, a first name, a last name or a nick name.

For instance, we found that *Bose* headsets and *Apple* smartphones broadcast such device names that include the name of the owner: `LE-Bose Alice` and `Bob's iPhone`. As those devices can be visually identifiable, a *passive* observer can then correlate information mined from the device name with visual observations to identify the individual. To make the matter worst, this privacy threat can be further extended leveraging online social networks such as *Facebook* [178], *Instagram* [179] and *Twitter* [180] where such devices names can leak the physical identity of the owner.

During our dataset anonymization process (see Section III.2.3), we identified and subsequently anonymized a total of 10.9% of the 1300 distinct device names advertised from 0.4% of the addresses of $dataset_{Passive}$.

IV.4 Active information inference

As part of the ATT protocol, discoverable BLE enabled devices expose a data structure called GATT profile describing supported features through concepts of *services* and *characteristics* (see Section I.2.2.4). Actually, this profile can be accessed by any *active* eavesdroppers in range and can expose users to privacy issues.

In this section, leveraging $dataset_{Active}$, we first shed light on several issues where such GATT profiles can be mined to infer sensitive information that can impact the privacy of users.

In a second time, we demonstrate how a Bluetooth harmless inherent request-response mechanism can have privacy implications. More specifically, we introduce a *timing attack* that can be triggered by an *active* attacker in order to infer the activity of a remote device.

Table IV.3 – Distribution of the top 10 services and characteristics in $dataset_{Active}$.

		Device addresses (%)		
		All	Stable	Private
Services	Generic Access	99.01	92.99	100
	Generic Attribute	97.34	81.35	99.97
	Apple Continuity	84.74	6.48	97.64
	Apple Nearby Service	84.32	4.36	97.50
	Device Information	69.74	55.90	72.02
	Battery Service	57.86	5.58	66.48
	Current Time Service	57.08	0.05	66.48
	Apple Media Service	57.07	0	66.48
	Apple NCS Service	57.07	0	66.48
	ISSC Transparent	3.86	27.26	0
	Overall*	99.17	94.16	100
Characteristics	Device Name	99.65	97.56	100
	Appearance	98.90	92.40	99.97
	Service Changed	97.34	81.35	99.97
	Apple Continuity	84.74	6.48	97.64
	Apple Nearby	84.32	4.36	97.50
	Manufacturer Name String	69.38	53.40	72.02
	Model Number String	69.32	52.98	72.02
	Battery Level	58.65	11.16	66.48
	Current Time	57.30	1.65	66.48
	Apple MS Entity Attribute	57.07	0	66.48
	Overall*	99.96	99.73	100

* Devices that include at least one of the top 10 service or characteristic.

IV.4.1 Observations on the dataset

As a reminder, our study is based on $dataset_{Active}$ which is constituted of BLE GATT profiles from 13295 distinct Bluetooth device addresses collected over five months (see Section III.2).

Note that, $dataset_{Active}$ was divided into two parts depending on the nature of the address used by devices (i.e. Stable or Private). In fact, devices using Private addresses can be observed several times under a different pseudonym. Therefore, in the Private part of $dataset_{Active}$, the number of actual devices is expected to be smaller than the reported number of distinct device addresses.

IV.4.1.1 Services and characteristics

Table IV.3 presents the distribution of the top 10 services and characteristics found within $dataset_{Active}$.

First, we can observe that `Generic Access` and `Generic Attribute` are the two most
included services with respectively 99.01% and 97.34% of GATT profiles. In fact, those
services contain generic information about the device such as its name and its appearance
that are carried by the `Device Name` and `Appearance` characteristics.

Afterwards, we witness custom services of manufacturers such as *Apple* and *ISSC*. From
our observations in Section III.4.2.4, we found that a large number of GATT profiles in
$dataset_{Active}$ are issued from *Apple* iPhone smartphones. As a result, this explains why
services and characteristics of *Apple* overpopulate $dataset_{Active}$.

Coming fifth with 69.74% of GATT profiles, the `Device Information` service embeds
manufacturer and/or vendor information about a device. Among its included characteristics,
we can cite the `Manufacturer Name String` and `Model Number String` which can be
privacy-invasive characteristics that we further study in Section IV.4.2.1.

Last but not least, as most connected devices are battery-powered, it is not surprising
that the `Battery Service` is part of our top 10 distribution. Indeed, leveraging such
a service, a device can expose the state of its battery through characteristics such as
the `Battery Level`. Similarly, many Bluetooth devices have the ability to store and
show time information. To this end, they can leverage the `Current Time Service` to
expose time information to other nearby BLE enabled devices through the `Current Time`
characteristic.

Finally, it is important to note that more than 99% devices include at least one of the
services and characteristics presented in our computed top 10 distribution. Provided that
they are readable without authentication, information such as device names, appearances,
manufacturer names, device models and battery levels can be trivially accessed by remote
attackers, and can raise privacy concerns (see Section IV.4.2).

IV.4.1.2   Readable characteristics

Table IV.4 presents an extended list of readable values from Bluetooth SIG defined
characteristics exposed in $dataset_{Active}$.

From this list, we can observe that both `Device Name` and `Appearance` are included in
more than 98% of the devices and are readable in about 99.5% of the cases. This is normal
as they are mandatory to implement and to be readable without authentication [38, Vol 3,
Part C, sec. 12].

A similar observation can be made on the `Service Changed` characteristic which is
mandatory [38, Vol 3, Part G, sec. 7] and shall be only used to indicate to connected
devices that services have changed (i.e. added, removed or modified).

Additionally, we can notice the presence of the `Manufacturer Name` and `Model Number`

Table IV.4 – Selection of Bluetooth SIG defined characteristics values that have been observed as readable at least once in $dataset_{Active}$. Percentages reported for readable values are fractions of devices for which this value is readable.

| Characteristic | Readable values | | | | | |
| | All | | Stable | | Private | |
	%	#	%	#	%	#
Device Name	99.49	13182	99.18	1821	99.54	11361
Appearance	99.48	13082	98.56	1714	99.62	11368
Manufacturer Name String	99.48	9177	99.40	999	99.49	8178
Model Number String	99.36	9158	99.40	991	99.36	8167
Software Revision String	97.04	1017	97.68	1010	50	7
Hardware Revision String	95.95	996	96.58	989	50	7
Serial Number String	97.12	979	97.79	972	50	7
Firmware Revision String	94.99	835	95.72	828	50	7
System ID	97.51	822	98.31	815	50	7
PnP ID	98.02	741	98.02	741	0	0
Battery Level	2.45	191	90.95	191	0	0
Current Time	0.41	31	100	31	0	0
Body Sensor Location	70	28	100	28	0	0
Sensor Location	100	8	100	8	0	0
Service Changed	0.02	2	0.13	2	0	0

`String` which are characteristics that both Stable and Private device addresses use to expose their manufacturers and model names.

`Software Revision String`, `Hardware Revision String`, `Serial Number String`, `Firmware Revision String`, `System ID` and `PnP ID` carry digital identifiers whose values are readable in more than 94% of GATT profiles including those characteristics. In addition, we can observe that `PnP ID` is a characteristic for which devices using Private addresses do not expose its values. As such a characteristic embeds a unique identifier, this can be explained by the fact that it cannot be compliant with the privacy protection brought by Private addresses. Also, beyond tracking concerns, we further demonstrate that values of those characteristics can lead an attacker to infer information such as the model and manufacturer of a device that can breach the privacy of its owner (see Section IV.4.2.2).

Finally, the `Battery Level`, `Current Time`, `Body Sensor Location` and `Sensor Location` are four characteristics that are only readable within GATT profiles of devices using Stable addresses. As for the `PnP ID` characteristic, this is likely due to privacy preserving choices as they can be useful material to defeat the anti-tracking feature [181, 68].

IV.4.2   Inferring information from GATT profiles

In this section, we present a number of elements found in GATT profiles that can be used to infer potentially sensitive information on the device and its corresponding user.

In particular, often readable without authentication (see Table IV.4), we found that the value of characteristics is a rich source of information.

To summarize our findings, we identified that information included in GATT profiles can be used to infer the device type, model, manufacturer, software version and the name of the user.

De facto, all those information can threaten the privacy of the device owner. Indeed, information on the device model can lead to *inventory attacks* [59], and the name of the user can reveal the identity of the owner (see Section IV.3.2.3). Furthermore, we found that the value of some characteristics can hold identifiers, which can be leveraged for tracking despite the device address randomization.

IV.4.2.1   Human readable identifiers

GATT profiles can include characteristics for which the value is a human readable string. For instance, this is the case of the `Device Name`, `Model Number String` and `Manufacturer Name String`.

The `Device Name` characteristic is available and readable in more than 99% of the profiles, and often includes names of *manufacturer*, *model* and *user* such as `Polar M400` and `Alice's MacBook Pro`. Note that, during our dataset anonymization process (see Section III.2.3), we identified and redacted a total of 14.2% of the 1336 distinct device names found within $dataset_{Active}$ that potentially contain the identity of the owner.

Similarly, values carried by the `Model Number String` explicitly identify the device *model*. For instance, the model number of an *Apple* iPhone 8 is `iPhone10,4` [182], while `iPad8,3` and `MacBookPro15,1` respectively indicate a 11-inch *Apple* iPad Pro and a 2018 15-inch *Apple* MacBook Pro [183].

Finally, the `Manufacturer Name String` contains a UTF-8 string that directly reveals the *manufacturer* of the device. Among manufacturers spotted in $dataset_{Active}$, we noted that *Porsche*, *Oculus* and *Apple* include the `Manufacturer Name String` characteristic.

IV.4.2.2   Digital identifiers

Serial number strings: The `Serial Number String` characteristic carries a variable-length UTF-8 string representing the serial number for a particular instance of the device. Leveraging $dataset_{Active}$, we found that the format of this identifier is often specific to a vendor and thus can be leveraged to infer the *manufacturer*. As an example, *Ultimate Ears* and *Bose* respectively code their serial numbers following the `"^1...LZ0....800$"` and `"^07....[Z,P][6,7,8]"` regular expressions.

System identifiers: The value of the `System ID` characteristic is a 64-bit structure which consists of a 40-bit manufacturer-defined identifier concatenated to a 24-bit OUI. By definition, the OUI is issued by the IEEE Registration Authority to companies and, as a consequence, can reveal the *manufacturer*. For instance, manufacturers such as *Xiaomi* and *Amazfit* include their OUI within the value of their `System ID` characteristic.

PnP identifiers: The `PnP ID` characteristic carries a set of values that are used to create a unique identifier for the device. Included in this characteristic is a `Vendor ID Source`, a `Vendor ID`, a `Product ID` and a `Product Version` field. The `Vendor ID Source` specifies the type of the `Vendor ID` value: a company identifier assigned by the Bluetooth SIG [171] or a value assigned by the USB Implementers Forum [184]. From our observations, manufacturers such as *Gigaset* (company ID `0x0180`) and *Freebox* (USB ID `0x10eb`) include an identifiable `PnP ID` in their GATT profiles.

Therefore, the `PnP ID` can reveal the *manufacturer* but also the device *model* and *software version*. For instance, `Product ID` of *Fitbit* Surge and *Fitbit* Charge are respectively `0x0010` and `0x0013`, while the `Product Version` of the *Bose* SoundSport Free earphones is `0x0132` and corresponds to the value carried by its `Software Revision String` (i.e. `1.3.2`).

Version identifiers: The `Software Revision String`, `Hardware Revision String` and `Firmware Revision String` characteristics include a UTF-8 string that respectively represents the version of the software, the hardware and the firmware of the device. To illustrate, we observed that devices of *Polar* and *Xiaomi* manufacturers include the `Firmware Revision String` characteristic disclosing their *firmware versions*.

Also, in addition to the tracking issues that those version identifiers can raise [1], they can be leveraged by eavesdroppers to gain knowledge on the OS of the targeted device prior to trigger an attack.

IV.4.2.3   Enumerated type values

Some characteristics are associated with enumerated values whose meanings are specified by the GATT specifications [47].

Appearances: The `Appearance` characteristic represents information about the external appearance of the device. Readable in more than 99% of GATT profiles, such a characteristic can provide a broad description of the device as well as a more specific one. As an example, appearance values of *Apple* TV and *Garmin* Forerunner 230 devices are respectively `Generic Media Player` and `Watch:  Sports Watch`.

Moreover, as reported in Section IV.3.2.2, certain appearances indicate specific medical

---

1. Because they are related to the device hardware, hardware revision strings are supposed to be more stable identifiers than the software and firmware revision ones.

Table IV.5 – List of `Body Sensor Location` values extracted from [2, sec. 3.24.2.1].

Value	Description
0x01	Chest
0x02	Wrist
0x03	Finger
0x04	Hand
0x05	Ear Lobe
0x06	Foot

Table IV.6 – List of `Sensor Location` values extracted from [2, sec. 3.152.2.1].

Value	Description
0x01	Top of shoe
0x02	In shoe
0x03	Hip
0x04	Front Wheel
0x05	Left Crank
0x06	Right Crank
0x07	Left Pedal
0x08	Right Pedal
0x09	Front Hub
0x0a	Rear Dropout
0x0b	Chainstay
0x0c	Rear Wheel
0x0d	Rear Hub
0x0e	Chest
0x0f	Spider
0x10	Chain Ring

devices such as an `Insulin Pen` that can trivially betray a medical condition of the owner.

Sensor locations: The `Body Sensor Location` and `Sensor Location` characteristics are indicating the position of the device on the user. As such, they provide clues to infer the *type* of device. Indeed, the presence of one of those characteristics indicates that the device is a sensor, and its value can further specify the type of sensor (see Table IV.5 and Table IV.6 for respectively a list of `Body Sensor Location` and `Sensor Location` values). Leveraging $dataset_{Active}$, we observed that manufacturers such as *Suunto*, *Geonaute* and *Mio Global* reveal the location of the device on its owner this way.

IV.4.2.4 Measurement values

Frequently standardized and expressed as the product of numerical value and unit, the measurement value is the value given by a measuring instrument or device. As such, it can thus reveal the current state of a device.

During our study, we discovered that *Mio Global* Alpha 2 smartwatches expose a readable value within their `Running Speed and Cadence Measurement` characteristic revealing the physical activity of the user (i.e. walking or running). As a result, such a characteristic can constitute an additional source of information that can be exploited to profile or physically identify a user [143].

IV.4.2.5   Names of services and characteristics

Beyond values carried by characteristics, names of services and characteristics can be leveraged as an indicator to reveal both the device *type* and *manufacturer*. For instance, the `Cycling Speed and Cadence Feature` characteristic will be only included in cycle devices, while the presence of the `Running Speed and Cadence Measurement` characteristic denotes running sensors.

In addition, we found that UUIDs of attributes exposed within GATT profiles can be customized by manufacturers. Leveraging online specifications and codes such as the *Apple* Notification Center Service (NCS) [185] and *Xiaomi* Mi Band 2 [186] ones, it becomes possible to uncover meanings of custom UUIDs, disclosing the corresponding *manufacturer* at the same time. As an example, the `7905f431-b5ce-4e99-a40f-4b1e122d00d0` and `00000006-0000-3512-2118-0009af100700` UUIDs are respectively associated with the *Apple* NCS service and *Xiaomi* Mi Band 2 battery characteristic.

IV.4.2.6   Misused characteristics

During our experiments, we observed that some characteristics can be misused: they carry data that are not related to their initial purpose.

In particular, through the *LG* G5 and *LG* G6 smartphones, we identified that the *LG* manufacturer exploits the `Altitude` Bluetooth SIG defined characteristic to include a static device address while using a Random Resolvable device address. As a reminder, we highlight that a static identifier advertised over random addresses can negate the anti-tracking feature.

Similarly, those devices leverage the `Location Name` and `HTTP Headers` characteristics to expose static UUIDs that constitute additional stable identifiers to undermine the LE Privacy provisions (see Section III.3.4.1).

Furthermore, although *LG* appears to be the only manufacturer to misuse Bluetooth SIG defined characteristics in $dataset_{Active}$, there could be other unobserved manufacturers that are making similar implementation misconceptions.

IV.4.3   Activity inference through a Bluetooth based timing attack

In this section, we introduce a timing attack that can be triggered by a nearby attacker in order to infer the activity of a remote Bluetooth device.

Especially, by observing timing variations of the ping mechanism of the L2CAP layer, we demonstrate that it is possible to detect state changes of a device, for instance when it goes in or out of the *locked* state. In addition, our experimental results show that the *change point detection* analysis of the timing allows to detect state changes with a high accuracy.

Finally, we briefly discuss the applicability of this attack to IoT devices.

IV.4.3.1   Device address acquisition

A prerequisite of our attack is the knowledge of the BD_ADDR of the targeted device. In fact, the attacker needs this information in order to send L2CAP ping requests to his target (see Section I.2.1). As a result, several techniques to obtain this device address are presented in the following.

Bluetooth inquiry scan: A Bluetooth device can be set in discoverable (as opposed to non-discoverable mode). Once in this mode, it will answer to inquiry signals and reveal its BD_ADDR. By performing an inquiry scan, the attacker can thus obtain the device addresses of nearby devices. Moreover, although it is supposed to be only used during pairing process, a number of Bluetooth devices are left in discoverable mode.

Bluetooth Upper Address Part (UAP) computation: By default, a Bluetooth device is in non-discoverable mode and does not respond to inquiry signals. Nevertheless, during communications with a previously paired object, Bluetooth frames are exchanged. While only the Lower Address Part (LAP) is available in clear as part of the header of those frames, an attacker can derive the UAP by leveraging information provided by the CRC and Header Error Check (HEC) fields [187]. As a result, by monitoring communications between two Bluetooth devices, the attacker can obtain the minimal (UAP, LAP) pair required to send L2CAP ping requests.

BLE Public address: By broadcasting advertisement packets, BLE devices advertise their presence. In practice, even if the LE Privacy feature has introduced random device addresses to reduce physical tracking concerns, many BLE devices still use their Public BD_ADDR (see Section III.3.2). Usually, the Public device address is shared between the BLE stack and all the other Bluetooth elements. To reach the L2CAP layer, such a Public address can be then leveraged. As a consequence, collecting Public BD_ADDR found within advertisement packets can reveal the device addresses of Bluetooth devices in range.

Correlated Wi-Fi MAC address: On mobile devices, Bluetooth and Wi-Fi can be embedded on the same chipset, and their device addresses are often contiguous [81]. As an illustration, the BD_ADDR of an *Apple* iPhone smartphone is mostly equal to its Wi-Fi MAC address plus or minus 1. To make the matter worst, the Wi-Fi MAC address is available in clear in many frames. Also, even if the device can hide its true MAC address using random MAC addresses in certain situations, Martin et al. demonstrated [81] that it can be still possible to obtain the original MAC address. Therefore, the attacker can exploit Wi-Fi to obtain the BD_ADDR of a device.

IV.4.3.2 Attack description

In the following, we describe a timing attack that uses the ping mechanism of the L2CAP layer (see Section I.2.1) to infer state changes of a Bluetooth enabled device. By measuring variations of the request-response round-trip time (RTT), this attack allows a nearby eavesdropper to identify when its targeted device goes through a change of state.

Given a device identified by its BD_ADDR, the attacker floods the L2CAP layer of the Bluetooth interface of his victim by sending `Echo Request` packets in large number. Each of those requests should trigger an `Echo Response` back. Then, the attacker measures the corresponding RTT and uses it to infer information on the status of the device.

The idea behind this attack is that the ping flood will stress the Bluetooth stack up to the L2CAP layer. The speed rate at which those ping requests are processed will depend on available resources of the target.

On mobile operating systems, available resources for the processing of those requests can vary depending on the smartphone state. For instance, those resources will be reduced if an application is running, or if the smartphone is in a power saving mode. As a consequence, the observations on the RTT can be exploited to gain information on the device state. By showing the variations of the RTT when the device goes alternatively in the *idle* and *locked* states, Figure IV.3 presents an illustration of this attack.

Using this approach, it is possible to run a *state change identification* attack in which the attacker can detect when the targeted device changes its state: for instance, when it goes from the *idle* to the *locked* state (see Figure IV.3).

IV.4.3.3 State change identification

Based on the measured RTTs, our attack aims at detecting state changes of smartphones.

Let $O = \{o_k\}$ be the sequence of observations (ping RTTs) and $T = \{t_k\}$ their corresponding timestamps. Similarly, let $C = \{c_i\}$ be the sequence representing the state changes: $c_i = 1$ if the device changed state at time t_i, $c_i = 0$ otherwise. As a result, our state change
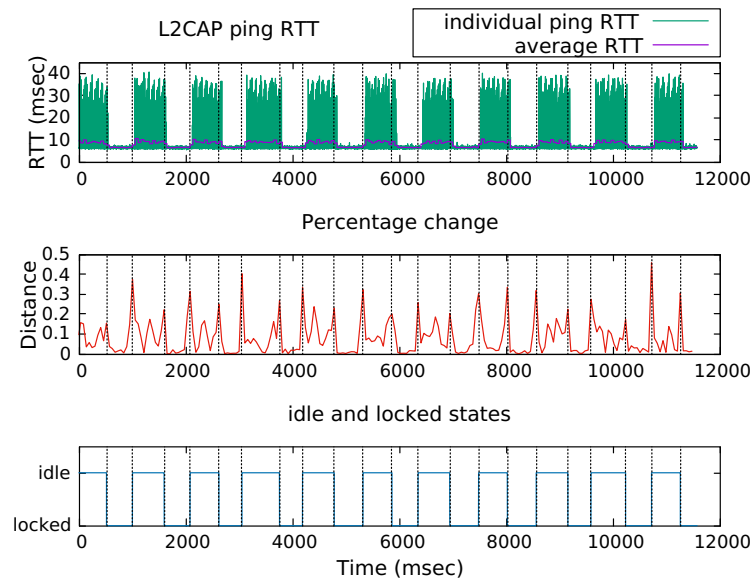
Figure IV.3 – Bluetooth ping flood RTTs (individual and average), percentage change and actual state (ground truth) while alternating between the *idle* and *locked* states. The percentage change is computed from the average RTTs. Vertical bars correspond to the times of the actual state changes.

identification problem falls down to inferring the sequence C based on the observation O.

As previously observed, the change of state is characterized by modifications of the measured amplitude of RTTs. Detecting amplitude changes in temporal series is a problem known as *change point detection* [188] for which several methods have been developed. Besides, one of them is the *percentage change* algorithm. Based on a sequence of temporal values, it outputs a series of percentage $P = \{p_i\}$ corresponding to the likelihood of a change at a given time: the higher the probability at a given point, the most likely the change.

Having this sequence of percentage changes, the next step is then to select the actual change points. This is done by setting a threshold value h and considering change points as all the time points for which the percentage change is above the threshold (i.e. time points t_i for which $p_i > h$).

For our study, we used `pct_change()`, the implementation of the percentage change of the `pandas` Python library [189]. Using the experimental approach detailed in Section IV.4.3.5, we then identified the threshold. In order to cope with the high variations of the RTTs, we rely on the average RTTs over a period of one second. To finish, Figure IV.3 shows the variation of the percentage change values as the device goes from a state to another.
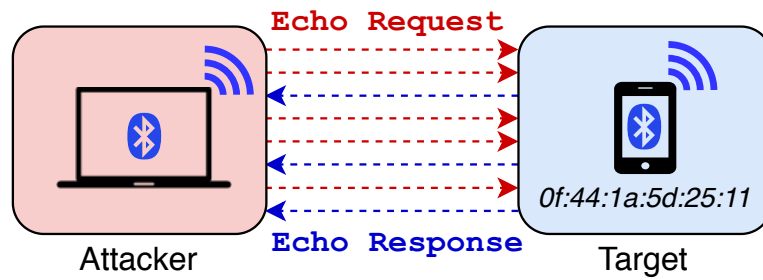
Figure IV.4 – Representation of the Bluetooth based timing attack. The attacker floods his target with L2CAP ping requests while measuring their corresponding RTTs. Note that, only `Echo Response` of B ping class are represented.

IV.4.3.4   Measurements of the L2CAP ping RTTs

Figure IV.4 describes our attack that requires to be able to send a large number of L2CAP ping requests at a regular interval, and to measure their corresponding RTTs. To this purpose, we based the implementation of our experimental tool on the native *Flood Ping* function of `l2ping` [190], a software available in `BlueZ` [133].

Actually, we modified `l2ping` to include several new features. As it waited for an `Echo Response` packet back before reissuing an `Echo Request`, a static code review along with a dynamic execution analysis revealed that the *Flood Ping* function of `l2ping` did not allow flooding. Thus, we removed this limitation by implementing concurrent threads capable of sending ping requests in parallel. Also, this allowed us to increase the request rate while ensuring that they were sent at a regular interval.

Then, we optimized the ping identifiers[1] (IDs) management to guarantee continuous RTT measurements and maintain a constant ping requests rate. Indeed, the identifiers used for measurements cannot be reused until the corresponding responses have been received. To bypass this issue, we interleaved two ping classes: A ping class requests that are only sent for flooding (ID $\in \{0\}$), and B ping class that are used for measurements as well (ID $\in [1; 255]$).

IV.4.3.5   Experimental evaluation

In this section, leveraging a set of three smartphones, we present the experimental evaluation of our Bluetooth based timing attack.

---

1. Used to match L2CAP `Echo Request` with `Echo Response`, we remind that the `Identifier` is 8-bit long (see Section I.2.1). As a consequence, such an `Identifier` can take 256 values ($\in [0; 255]$) implying that only 256 active pings can be handled in parallel.

Table IV.7 – List of tested devices

Device	OS	Bluetooth version
Apple iPhone 4	iOS 7.1.2	2.1
Apple iPhone 6	iOS 11.0.3	4.2
Samsung Galaxy A3	Android 5.0.2	4.0

IV.4.3.5.1 Considered devices and states

Tested devices: Experiments have been performed using three devices listed in Table IV.7. Those devices cover both Android and iOS, the two most popular mobile OS, as well as older (*Apple* iPhone 4) and more recent devices (*Samsung* Galaxy A3 and *Apple* iPhone 6). *Apple* iPhone 4, the oldest smartphone in our set of devices, has an early Bluetooth version. As all Bluetooth versions implement the L2CAP ping mechanism, it shows that the attack can be efficient on old Bluetooth versions as well.

Device states: Throughout our experimental study, we focused on a set of states that are representative of the standard usages of a smartphone. First, we considered *locked*, *idle* and *active* as states in which a device is often set. Thereafter, we considered a popular application (*Shazam*) to represent the state when the device is running an application. Moreover, we selected another application (*BLEScanner*) which actively uses the Bluetooth interface of the device. Finally, we also considered two states (*BT inquiry* and *Wi-Fi Scan*) in which the device uses its Bluetooth or Wi-Fi wireless interface.

To summarize, the seven considered device states are the following:

— *locked*: the phone is left untouched after its screen has been turned off;
— *idle*: the screen is on, and no application [1] is running neither in the foreground nor in the background;
— *active*: the screen is on, no application [1] is running neither in the foreground nor in the background, and the user only swipes his finger over the screen;
— *Shazam*: the Shazam [191] application is running in the foreground;
— *BLEScanner*: the BLEScanner [192] application is running in the foreground;
— *BT inquiry*: an inquiry scan for nearby discoverable Bluetooth devices is performed in the foreground;
— *Wi-Fi Scan*: a scan for nearby Wi-Fi networks is performed in the foreground.

IV.4.3.5.2 Experimental protocol

To evaluate the performances of our approach as well as the potential differences between the various states, we conducted series of measurements. For each pair of the states listed above: we alternatively put the smartphone in each of the state for 10 seconds for an

---

1. Apart from system specific applications.

overall session of 120 seconds corresponding to a total of 12 changes.

As the transition between the states requires to go through one intermediary state, some pairs were not possible. For instance, it is not possible to go directly from the *Shazam* state to the *BT inquiry* one as it is necessary to go through the *active* state first. While performing those manipulations, the device was submitted to active L2CAP ping measurements and the time of transitions were manually noted. The output of each experiment is a set of RTT measurements and a set of timestamps corresponding to changes of states.

Note that, as the state change monitoring was done manually, it does not have the same accuracy as the RTT measurements timestamps. Thus, we considered the granularity of time change detection of one second, which corresponds to the manual measurement expected accuracy.

Based on those measurements, it is possible to evaluate the state change identification attack. The performance evaluation has been done using the following metrics: *True Positive Rate* (TPR): the ratio of correct inference, *False Positive Rate* (FPR): the ratio of incorrect inference. In the case of the state change identification attack, a true positive is the detection of an actual change of state, while a false positive is a detection of a state change where there is none.

Flooding parameters: We want to maximize the stress on the Bluetooth stack while maintaining regular measurements. In particular, we want to avoid Bluetooth requests being dropped because the load was too high. Through empirical experiments, we found that sending a request every 6 milliseconds was producing good results in term of RTT variations. To keep a sufficient granularity of the measurements, we set the interval for B ping class requests to 18 milliseconds.

IV.4.3.5.3   Results

In the following, we focus on the results obtained with the *Apple* iPhone 6 device. In fact, its recent Bluetooth version as well as its monitoring ease with low generated noise makes this smartphone the ideal candidate to demonstrate our attack.

The efficiency of our state change identification technique has been evaluated using the previously described experimental protocol. Figure IV.5 presents the Receiver Operating Characteristic (ROC) curves of this detector for state changes either from or to the *idle* state.

From those curves, we can assert that our approach is able to detect those changes with a high accuracy. As an example, it reaches a TPR of 0.917 for a FPR of 0.046. Overall, the Area Under the Curve (AUC) is high and close to 1. Note that, similar performances are observed for all cases except for the *active* state, which is close to the *idle* state as the only practical difference is that the user swipes his finger over the screen.
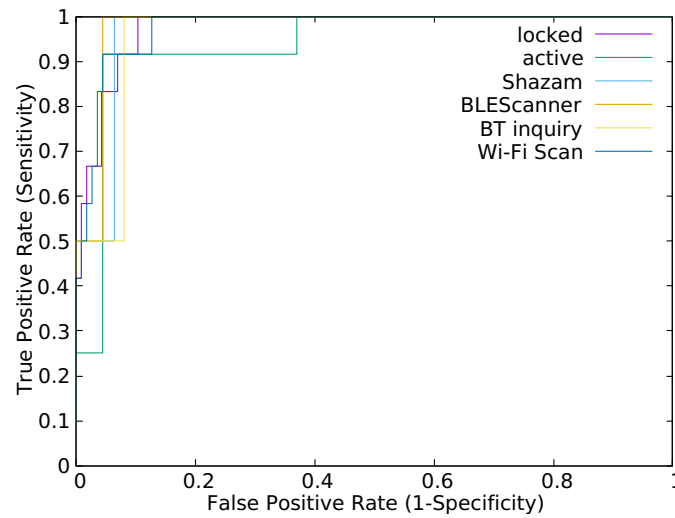
Figure IV.5 – ROC curves for the state change identification from the *idle* state with the *Apple* iPhone 6 smartphone.

Table IV.8 – *Apple* iPhone 6 state change identification with AUC values.

From ↓ to →	idle	locked	active	Shazam	BLEScanner	BT inquiry	Wi-Fi Scan
idle	–	0.976	0.939	0.968	0.978	0.960	0.975
locked	0.988	–	0.960	0.964	0.973	0.967	0.981
active	0.931	0.989	–	0.965	0.966	0.973	0.974
Shazam	0.962	0.944	0.958	–	N/A	N/A	N/A
BLEScanner	0.987	0.971	0.978	N/A	–	N/A	N/A
BT inquiry	0.942	0.956	0.964	N/A	N/A	–	N/A
Wi-Fi Scan	0.959	0.967	0.966	N/A	N/A	N/A	–

Table IV.8 summarizes results of the state change identification for the *Apple* iPhone 6 smartphone by showing the AUC for each state transition. As shown for the *idle* case, AUCs are high and all above 0.9 demonstrating the good performances of the state change detector.

Comparable performances were obtained on the two other smartphones. Indeed, for the *Apple* iPhone 4, AUC $\in [0.922; 0.980]$ where the lowest and highest values respectively represent *Wi-Fi Scan* to *locked* and *locked* to *BT inquiry* states transitions (see Table E.1 in Appendix E).

On the other hand, the attack was more difficult to perform on the *Samsung* Galaxy A3 as it only accepts ten thousand `Echo Request` before to reset the L2CAP connection. In this case, we kept the experimental protocol with a lower overall session of 60 seconds corresponding to a total of 6 changes. AUC $\in [0.913; 0.985]$ represents the results with this smartphone where the lowest and highest values respectively correspond to *BLEScanner* to *locked* and *Wi-Fi Scan* to *idle* states transitions (see Table E.2 in Appendix E).

IV.4.3.6 Applicability to IoT devices

Although the experimental evaluation has been only performed on smartphones, all Bluetooth devices that support this request-response mechanism of the L2CAP layer are potentially threatened. As a consequence, IoT devices can be also subject to this attack.

To aggravate the situation, by taking advantage of those single-task connected objects, the attacker can define the action that is taking place by analyzing variations of RTTs. For instance, variations of RTTs in a Bluetooth e-cigarette [193] can allow the attacker to know when his target is smoking without having previously fingerprinted different states of this device.

IV.5 Implementation: *Himiko*, a *Human Interface for Monitoring and Inferring Knowledge on Bluetooth Low Energy Objects*

In this section, we introduce *Himiko*, a *Human Interface for Monitoring and Inferring Knowledge on Bluetooth Low Energy Objects*. In fact, this tool aims to extend our work by capturing and processing in real-time the content of advertisement packets and GATT profiles.

Based on the online official Bluetooth SIG resources [194] along with our research findings (see Section IV.3.2 and Section IV.4.2), we implemented a Python parser that can decode both structured (specified) and manufacturer specific (non-specified) data. In addition, we also provide a user-friendly interface to display the extracted information.

IV.5.1 The *Himiko* tool

The *Himiko* tool is based on a three-step process. First, advertisement packets emitted by a BLE enabled device are captured through a Bluetooth interface. Then, the raw data are parsed and processed to extract information that can both defeat the randomization mechanism (see Section III.3.4) and be used to infer attributes of the owner. Finally, results of the processing are displayed as a feedback to the user. The architecture of the tool is presented in Figure IV.6.

To capture the advertisement packets, *Himiko* only requires two Bluetooth cards supporting the BLE protocol. This is the case for most basic off-the-shelf Bluetooth cards on modern computers running a Unix-like system. Devices in range are detected using the RSSI, and leveraging external dongles such as a CSR v4.0 Bluetooth USB dongle can simplify the proximity estimation of users devices.
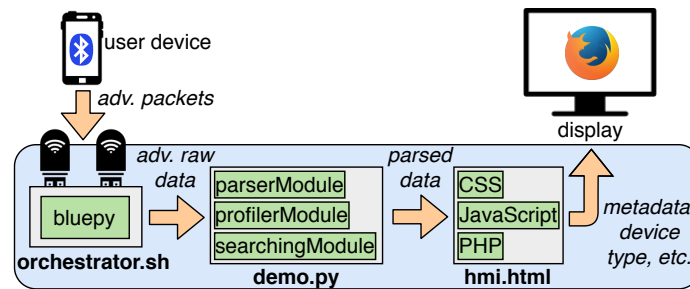
Figure IV.6 – Architecture of the *Himiko* tool.

Actually, *Himiko* is composed of three main files. The first one, `orchestrator.sh`, is a Bash script configuring the Bluetooth interfaces to enter in the BLE scanning mode featured by the `bluepy` [134] Python library. The latter outputs the captured advertising raw data in a string stream, which can be then parsed and processed in real-time by the second Python script, `demo.py`.

This second script is composed of three Python modules – `parserModule`, `profilerModule` and `searchingModule` – that respectively parse the advertising raw data, fetch the GATT profile of the device and search for this device on *Google* Shopping [173] and *Qwant* [110] online marketplaces. Finally, `hmi.html` embeds web technologies (i.e. CSS, JavaScript and PHP) that structure and beautify results of the analysis before to be displayed in a *Mozilla* Firefox [195] web browser.

The knowledge base used by `parserModule` to decode advertisement packets and GATT profiles has been built from the online official Bluetooth SIG resources, third-party public resources[1] and the reverse engineering of the *Apple* Handoff, *Apple* Nearby Info, *Microsoft* CDP and *Google* Nearby proximity protocols (see Section III.3.4.2).

Note that, the only information captured by *Himiko* are data contained in advertisement packets and GATT profiles of BLE devices having an enabled Bluetooth interface. As a consequence, traffic data sent by associated devices and timing or physical-layer information are not considered.

The display includes the metadata of the advertisement packet (i.e. BD_ADDR, type of BD_ADDR, etc.), a list of the extracted information, an inference of the device type based on its advertised device name, and a dump of the GATT profile of the device. In addition, we point out that a GATT profile is readable by default as the Bluetooth Core Specification [38, Vol 3, Part G, sec. 8.1] does not consider included information as private or confidential.

Finally, when a BLE device broadcasts its device name, the owner of the involved device

---

1. The `advlib` [137] advertisement packet decoding library, the `RaMBLE` [136] Android mobile application and the `bleah` [135] BLE scanner tool.

can decide to provide additional information [1] on the device in order to improve the device identification capabilities of *Himiko*.

Also, in order to minimize privacy risks for the users of *Himiko*, we apply the minimization principle and keep as little necessary information as possible. In particular, all collected data are kept in memory during the processing time before being immediately erased when the results are displayed to the users. Furthermore, the tool only detects devices in close range of the antenna [2] to ensure that only volunteering participants will have their data collected and processed.

IV.5.2   Interaction with participants

During the demonstration, participants are able to interact with *Himiko* by testing the information broadcasted by their BLE enabled devices. By bringing their device close to the Bluetooth antennas of the tool, they will trigger a capture event that will record an advertisement packet emitted by their device. Such a packet contains raw data that will be parsed and processed to compute a comprehensive analysis of the emitted information of the device.

Results of this process are then displayed on a screen as a feedback to the user. As an example, Figure F.1 in Appendix F presents the output triggered by a *Chipolo* Classic BLE keyring device. Moreover, as previously mentioned in Section IV.5.1, when a device advertises its device name, participants will have the opportunity to contribute to the knowledge base of *Himiko* by providing additional information on their devices.

IV.6   Recommendations

In this section, in light of the presented active information inference issues, we provide recommendations that should be considered by manufacturers of Bluetooth/BLE enabled devices, but also by the Bluetooth SIG to improve the Bluetooth Core Specification.

Restricting access to values of characteristics included within GATT profiles: During our work, we found that a number of characteristics are readable without authentication, and expose the device to both fingerprinting (see Section III.4) and inference of sensitive information (see Section IV.4). In many cases, it is not clear why this information is left openly available.

A simple solution should be to use the permission system of GATT to ensure that those values can be only read by authenticated devices (see Section I.2.2.4). To be efficient,

---

1. Assisted by the operator, the owner can provide a regular expression matching the device name along with information such as the vendor, model and type.

2. A few centimeters.

Table IV.9 – Empirical entropy of characteristics exposed within GATT profiles of $dataset_{Active}$, without values (i.e. only handles, UUIDs and properties are considered).

		Entropy (bits)		
		All	Stable	Private
Characteristics ⟨Hdle, UUID, Prop⟩	Device Name*	1.913	1.191	0.731
	Appearance*	1.148	0.625	0.578
	Service Changed*	0.766	0.566	0.290
	Apple Continuity	0.680	0.317	0.462
	Apple Nearby	0.720	0.306	0.499
	Manuf. Name String	1.372	0.517	1.031
	Model Number String	1.327	0.491	1.030
	Battery Level	0.985	0.361	0.879
	Current Time	0.886	0.289	0.866
	Apple MS Entity Att.	0.835	0.277	0.831
	Overall	2.764	1.208	1.564
	Overall (services + characteristics)	2.765	1.208	1.564

* Value of this characteristic is kept during the entropy computation because the Bluetooth Core Specification specifies it as mandatory.

this mechanism should be set on all characteristics by default, and its removal should be justified by a valid requirement.

In order to simulate the adoption of this measure, we removed all values of characteristics within $dataset_{Active}$ at the exception of mandatory ones (i.e. values of the `Device Name`, `Appearance` and `Service Changed` characteristics). Afterwards, we re-evaluated the entropy and found that removing those values has a significant impact on the fingerprinting potential: the overall entropy goes down from 4.380 (see Table III.7) to 2.765 bits (see Table IV.9).

In cases where this solution cannot be adopted, values should be as general as possible. As an example, the value of the `Model Number String` characteristic within *Apple* devices could just be `Apple` or `iOS` instead of `iPhone10,4` or `iPad8,3` (as reported by Martin et al. [1]). Note that, we believe that this recommendation has to be addressed by manufacturers at the kernel or firmware level.

Minimizing exposure of GATT profiles: The current version of the Bluetooth Core Specification [38, Vol 3, Part G, sec. 8.1] stipulates the following: *"the list of services and characteristics that a device supports is not considered private or confidential information, and therefore the Service and Characteristic Discovery procedures shall always be permitted"*. Nevertheless, we demonstrated that this is not the case.

Indeed, even if values are not readable, the list of services and characteristics available in a GATT profile can be leveraged for fingerprinting and inference of sensitive information. Furthermore, we showed that the exhibition of metadata such as names of services and

characteristics is enough to leak sensitive information of users (see Section IV.4.2.5).

As a consequence, a potential mitigation technique would be to minimize the exposure of GATT profiles. This could be done by setting access control properties to services and characteristics so that only authenticated devices can access them.

By default, an unauthenticated device will then uniquely see basic services and characteristics (e.g. `Generic Attribute` and `Service Changed`), and only authenticated devices will be able to see the full list. For instance, a Peripheral that solely allows the reading of its `Battery Level` characteristic value to authenticated devices should hide the presence of the corresponding `Battery Service` during the service discovery procedure for unauthenticated devices. To our mind, this recommendation has to be addressed by the Bluetooth SIG as an erratum to the Bluetooth Core Specification.

Preventing the activity inference through the Bluetooth based timing attack:
The presented timing attack (see Section IV.4.3) is practical because any Bluetooth hosts in range can exchange L2CAP ping requests at a high rate. Therefore, a first naive countermeasure should be to limit the rate at which a remote host can perform those actions. Likewise, flood detection and blacklisting mechanisms can be supplementary protections that manufacturers could implement in order to undermine this attack.

Additionally, from our experiments, we attested that the accuracy of the timing measurements is also a key element of the attack. Thus, introducing a perturbation by enforcing a random back-off time in the response process could reduce its efficiency as well.

To finish, although Bluetooth features mutual authentication mechanisms, we showed that the access to L2CAP services is not limited to authenticated hosts. Hence, restricting such access exclusively to authenticated hosts could be another way to prevent this kind of attack.

IV.7   Conclusion

To conclude, this chapter aimed to demonstrate how *passive* and *active* attackers can infer information on users (i.e. their attributes, ongoing activities, etc.) from wireless communications of their carried Bluetooth/BLE enabled devices.

First, we detailed our threat model that considers devices including privacy issues in data exposed within advertisement packets and GATT profiles. In particular, we defined a successful[1] attack as an eavesdropper being able to learn information on nearby users, either through *inventory attacks* or *activity inference*. As a side note, we draw the attention to the fact that we limited our model to information that can be directly inferred from the observations of exposed data.

---

1. Beyond the term *successful*, we described *means* to infer users information.

Second, relying on our described threat model, we demonstrated that advertising data hold enough information to disclose the manufacturer, model and type of a device, that can betray personal characteristics as well as the medical condition of its corresponding owner. Especially, we showed that such devices information can be inferred from advertising data contained in the `Appearance` and `Manufacturer Specific Data` AD structures, but also from service UUIDs and Public BD_ADDR. For instance, the `Appearance` code `0x0d48` denotes an insulin pen, while the `0000de00-3dd4-4255-8d62-6dc7b9bd5561` service UUID is broadcasted by *Nikon* cameras.

Furthermore, when advertising data include names of users, we explained how an attacker can leverage visual observations to match the physical identity of the individual with its network identifier (i.e. its BD_ADDR). As an illustration, *Bose* headsets and *Apple* smartphones leak names of their owners through their advertised devices names.

As for Chapter III, on April 5th, 2019, we informed the manufacturers [1] of the identified privacy issues. However, as of July 7th, 2020, none [2] of those issues have been addressed.

Third, we analyzed data exposed within GATT profiles of BLE enabled devices. Leveraging $dataset_{Active}$, we first found that human readable and digital identifiers embedded in characteristics such as `Device Name`, `Appearance`, `Manufacturer Name String`, `Model Number String` and `Software Revision String` can be mostly accessed (i.e. read) without authentication. Then, we described how they can be mined to infer information on the device such as its type, model, manufacturer and software version. Similarly, we pointed out that metadata such as names of services and characteristics can be leveraged to threaten the privacy of users as well.

Besides, during this work, we discovered that the *LG* manufacturer diverts several Bluetooth SIG defined characteristics for other purposes. In fact, we found that the *LG* G5 and G6 smartphones carry static identifiers within characteristics such as `Altitude` while using Random Resolvable device addresses. In addition to the physical tracking concerns that this misuse can raise, this finding assesses the lack of control with regard to the release of connected devices.

Fourth, we presented a Bluetooth based timing attack that exploits a request-response mechanism of the L2CAP layer in order to infer the activity of a smartphone. More specifically, we demonstrated that combining timing measurements with a change point detection analysis allow to detect a device state changes with a high accuracy.

To begin this study, we provided four methods aiming to obtain a Bluetooth device address, a prerequisite for this timing attack. To exacerbate the situation, we identified that three of those methods acquire this device unique identifier when it is in non-discoverable mode. Because of the tracking and privacy issues that can result, a device address must be

---

1. Notifications have been sent to: *AirBolt*, *Apple*, *Bang&Olufsen*, *Bose*, *Diggro*, *Fitbit*, *Garmin*, *Giant*, *Google*, *GoPro*, *Jabra*, *Jewelbots*, *Logitech*, *Microsoft*, *Nikon*, *Nintendo*, *OculusVR*, *Samsung*, *Sony* and *Tile*.

2. To the best of our knowledge.

difficult to obtain, and more particularly when a device is in a mode that hides it from any other devices in range.

To pursue our work, we described the attack and the set of three Bluetooth featured smartphones on which it has been tested. Then, we explained how to detect changes between seven distinct states that have been chosen to be representative of the standard usage of a smartphone. From the experimental results, we attested that a remote *active* attacker can easily get information on the state of a device (i.e. idle, locked, etc.), that can be further exploited to obtain additional information on its user. As an example, the state change identification can be correlated with visual observations in order to identify a user manipulating his device.

Fifth, in light of the reported privacy issues, we introduced *Himiko*, a *Human Interface for Monitoring and Inferring Knowledge on Bluetooth Low Energy Objects*. Beyond its user-friendly interface, we put efforts into implementing this tool to 1) raise public awareness with regard to the information that can be disclosed within BLE enabled devices and 2) alert on the need to complement the Bluetooth Core Specification with additional requirements that would cover privacy issues on the BLE protocol. In fact, even if the Bluetooth Core Specification includes privacy considerations, they remain limited to the management of random link layer identifiers. Nevertheless, we showcased other elements of the advertising mechanism that can be used to breach privacy.

Lastly, we provided a set of recommendations to limit the impact of the outlined privacy threats. To our mind, the most straightforward solution is either to remove the unnecessary information that are being leaked in clear or set appropriate security permissions on them (e.g. allowing the reading of characteristic values only to authenticated devices). In addition, the amount of data exposed during services and characteristics discovery procedures of GATT profiles should be kept to a bare minimum to avoid fingerprinting and inference of users information.

Likewise, we specified three potential countermeasures in order to protect devices from the activity inference through the Bluetooth based timing attack. Note that, those countermeasures range from the naive implementation of flood detection and blacklisting mechanisms by manufacturers to more complex protections such as the enforcement of a random back-off time in the ping response process.

Chapter V

The case of personal data leaks in *Apple* BLE Continuity protocols

This chapter deals with personal data leaks applied to the case of Apple BLE Continuity protocols. At first, Apple Continuity protocols and their applications are introduced in Section V.1. Afterwards, Section V.2 details our thorough reverse engineering of such continuity protocols. Leveraging identifiers and counters included in Apple Continuity messages, passive tracking concerns are demonstrated in Section V.3. Section V.4 discusses a new active tracking threat based on the replay of corrupted Handoff messages. Exposed device status and characteristics are listed in Section V.5. Section V.6 shows that messages broadcasted by HomeKit accessories can betray human activities in a smarthome. A method to recover hashed e-mail addresses and phone numbers contained in messages of AirDrop and Nearby Action protocols is presented in Section V.7. Section V.8 exploits passive observations on the BLE advertising traffic to infer Siri voice commands of a remote user. The impact of the reported attacks with regard to their feasibility and privacy risk that can result from collected information is evaluated in Section V.9. Section V.10 provides recommendations that should be addressed by Apple to undermine the identified privacy vulnerabilities. At last, the conclusion of this chapter is given in Section V.11, while Section V.12 furnishes additional remarks.

Corresponding contribution : [3]

V.1   Introduction

Smart devices interacting with each other are bringing new types of applications[1] that simplify configuration procedures and enhance the user experience. To enable those features,

---

1. For instance, sending a file to a nearby device, transferring an activity to another device and sharing a network connection.

major vendors have developed protocols: *Google* Nearby [149], *Microsoft* CDP [148] and protocols used by *Apple* Continuity [196].

Besides, *Apple Continuity protocols*, the prominent family of protocols developed by *Apple*, can be found in all *Apple* products but also in devices from third-party companies [1]. As a consequence, such protocols are embedded in more than 1.5 billion active devices [199], including smartphones, laptops, earphones, smartwatches and smarthome appliances.

To enable services such as activity transfer, remote printing and smarthome monitoring, *Apple* Continuity protocols rely on BLE. Indeed, messages are carried by advertisement packets that are broadcasted over the air and thus made available to all devices in range.

In practice, we previously witnessed that wireless communications functionalities of smart devices can represent privacy threats. More specifically, Bluetooth/BLE and Wi-Fi signals can be leveraged for tracking users [77, 123] and inferring other private attributes [82, 200, 143].

To remedy to the tracking issue, we shed light on the LE Privacy feature [38, Vol 3, Part C, sec. 10.7] that defines the use of temporary and random link layer identifiers. Nevertheless, several works [59, 97] have disclosed privacy menaces associated with BLE jeopardizing the privacy of users despite the LE Privacy provisions. Furthermore, serious issues have been discovered [1] in *Apple* Continuity protocols, allowing an attacker to track a device through *passive* and *active* attacks.

In this chapter, we pursue on the path started by Martin et al. [1] and present a collection of new privacy concerns in *Apple* Continuity protocols. Based on a detailed reverse engineering of *Apple* Continuity protocols, we demonstrate how cleartext information exposed in BLE advertisement packets can be exploited for physical tracking but also to reveal personal information and users activities.

First, we present continuity protocols and describe the methodology that we used to provide an extensive reverse engineering of *Apple* Continuity protocols along with the format of their corresponding messages. Likewise, we discuss general features of such messages emphasizing that some of them are advertised over stable device addresses, trivially exposing their users to tracking.

Second, we identify new fields included within *Apple* Continuity messages under the form of identifiers and counters that can be exploited for *passive* tracking attacks. Among those fields, we found that the `Lid Open Count` as well as battery levels of AirPods earphones can be leveraged together to track their corresponding owners. Thereafter, we outline an additional privacy threat in which the global energy of the system could be fingerprinted to improve the amount of identifying information.

Third, we present a novel *active* attack that can be both used to track individuals and link devices belonging to the same iCloud account. Based on the replay of corrupted

---

1. *Apple* certified vendors [197] and manufacturers of HomeKit accessories [198].

Handoff messages, this attack supplements the one introduced in [1] which relies on Instant Hotspot messages in order to force a device to reveal its presence and identity. Moreover, we underline the fact that our attack is easier to exploit as Handoff messages are emitted by a wider range of devices, and therefore are more common than Instant Hotspot ones.

Fourth, we compile a list of several fields exposing characteristics of the device (e.g. category, model, OS version, color, etc.) and information on its status that can be used to infer the ongoing activity. Being advertised in clear, we then detail how those fields can be leveraged as a first step toward four possible privacy invasive applications, namely inventory attacks, visual identification, event correlation and activity monitoring.

Fifth, we show that messages broadcasted by HomeKit accessories include a `Global State Number` (`GSN`) that can leak human activities in a smarthome. In particular, we demonstrate that monitoring the increment of such field betrays a state change that is induced by the user, either in reaction to an environment modification (e.g. the user passes by a motion sensor and its detection causes the increment of the `GSN`) or because he executes a particular command (e.g. the user turns on/off its connected lightbulb). Throughout this study, we monitor an *Eve* motion sensor as well as an *Osram* lightbulb, and highlight that applying this attack to the whole range of HomeKit enabled devices (e.g. door sensors, thermostats, faucets, etc.) can reveal accurate information on smarthome activities.

Sixth, we discover that AirDrop and Nearby Action messages contain hashed e-mail addresses and phone numbers. Through *guesswork* attacks, we describe how they can be recovered in a matter seconds when the set of potential identifiers is constituted of several thousand entries. In addition, according to the performances of our simulation, we observe that this kind of attack is practical as the worst-case scenario (i.e. all the existing e-mail addresses) can be tested within an hour.

Seventh, we find that "Hey Siri" messages include a perceptual hash of voice commands and present how it can be exploited to infer a spoken command of a remote user. To begin this work, we introduce the *Philips Robust Hash* (PRH) algorithm and make some observations on the generation of the perceptual hash of Siri. Afterwards, we show how to build a dictionary of hundred commands and digests that will further serve to evaluate our dictionary attack on perceptual hashes. From the experimental results, we obtain a precision as high as 67% with a recall of 52% for an exact match of the perceptual hash.

Last but not least, we provide a discussion on the impact of the reported attacks. Relying on several elements such as the feasibility of such attacks with regard to the range, the context in which they can be performed and the privacy risk resulting from the information that an attacker can collect, we point out that the majority of exhibited attacks are easy to implement, and raise serious privacy leaks at the same time. Furthermore, we draw the attention to the fact that they can be mostly triggered by a passive attacker.

Finally, we furnish four potential solutions to mitigate the identified issues. Complementing the ones proposed in [97] and [1], we specify that the described countermeasures should

be addressed by *Apple* during the design and implementation of continuity protocols to preserve the privacy of their users.

V.2 Reverse engineering of *Apple* Continuity protocols

In this section, we first provide insights on continuity protocols prior to describe the methodology that we used to reverse engineer the suite of *Apple* Continuity protocols. Also, we present general features of *Apple* Continuity messages such as their content protection and the use of random device addresses.

V.2.1 Continuity protocols

Continuity protocols are network mechanisms that enable short range communications for services running on mobile devices and connected appliances. As such, they constitute the network element of *Apple* Continuity services [196], and similar protocols have been developed by major vendors: *Google* Nearby [149] and *Microsoft* CDP [148]. Most of the time, those protocols are leveraged to automatize configuration procedures through proprietary message structures. As a result, typical continuity services include activity transfer, multimedia content streaming, file sharing, device pairing, monitoring and control of smarthome accessories to name a few.

In a traditional Ethernet network, all communications go through a router that processes and delivers each data packets to the correct destination. On the contrary, continuity protocols do not need a central device and thus can improve throughput, energy efficiency and delay.

Basically, continuity protocols rely on wireless technologies such as Bluetooth/BLE and Wi-Fi to carry information between devices. As they are directly used by applications, such protocols can be seen as part of the *application* layer of the Open Systems Interconnection (OSI) model. Nevertheless, they sometimes include features (i.e. sequence number and reconnection) of other layers such as the *session* and *transport* layers. Because they only involve device-to-device (D2D) communications, the *network* layer is limited to a bare minimum, and both the *data link* and *physical* layers are handled by the underlying wireless protocol (e.g. Bluetooth, BLE or Wi-Fi).

V.2.2 Methodology

In this chapter, the presented analysis required an in-depth understanding of *Apple* Continuity protocols for which most specifications are not public. As a consequence, we

had to rely on supplementary sources of information, namely BLE traffic traces, outputs of *Apple* debugging tools and disassembled binaries.

Although limited in details and quantity, it is important to mention that there exist some official documents about *Apple* Continuity protocols and associated services. More specifically, we can cite the *Apple* Platform Security guide [201] and the HomeKit Accessory Protocol (HAP) specifications [202].

As *Apple* Continuity messages are embedded in BLE advertisement packets, the capture and generation of those packets were at the core of our study. In this regard, the capture was done leveraging our custom sniffer implemented over the `bluepy` [134] Python library, while a Bash script based on `hcitool` [203] was used to generate advertisement packets. In both cases, we used a CSR v4.0 Bluetooth USB dongle as transceiver. In order to isolate signals from a device, we placed the monitoring system next to it and filtered out packets coming from other devices based on their RSSI.

For the experiments, all the tests were performed in our laboratory against our devices and the ones owned by our institutions. As a result, those tests involved a range of *Apple* devices including iPhone smartphones, iPad tablets, AirPods earphones, MacBook laptops and Watch smartwatches that were running across different versions of *Apple* iOS, macOS and watchOS operating systems. In addition, we considered products from partner companies too that are compatible with the *Apple* ecosystem such as the *Eve* Motion sensor and the *Osram* Smart+ lightbulb.

Likewise, we relied on two *Apple* debugging tools: `PacketLogger` and `PacketDecoder`. Available on the *Apple* Developer website [204], those tools provide information on the content of received BLE frames, as well as on steps in the corresponding protocol.

Finally, we had recourse to disassembly in order to gain additional knowledge on *Apple* Continuity protocols. In fact, we used the `Hopper` [205] software to disassemble several macOS binaries such as `PacketLogger`, `PacketDecoder`, `CoreSpeech`, `HomeKit Accessory Simulator` and `sharingd`. Based on the disassembled codes, we were then capable of identifying the precise format of continuity messages, and the signification of some codes (e.g. device model, activity level, action type, etc.). Also, we were able to analyze the implementation of some functionalities such as the perceptual hash of Siri voice commands (see Section V.8).

V.2.3   General features of *Apple* Continuity messages

To transmit data to nearby devices, *Apple* Continuity protocols use the `Manufacturer Specific Data` AD structure of BLE advertisement packets. Starting with two bytes that indicate the company identifier of *Apple* (code `0x004c`), this AD embeds one or several *Apple* Continuity messages, accounting for up to 27 bytes. Outlined in Figure V.1, those *Apple* Continuity messages follow a TLV format (`Type` and `Length` are encoded on one
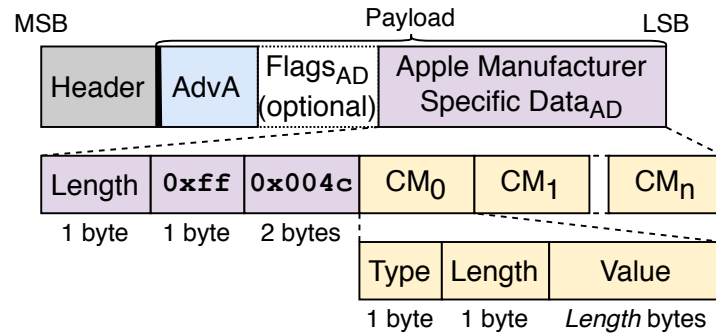
Figure V.1 – Structure of a BLE advertisement packet used to carry data of *Apple* Continuity protocols. The data are stored in a `Manufacturer Specific Data` AD structure (`0xff`) which starts with the company identifier of *Apple* (`0x004c`), followed by one or several continuity messages (CM) presented as a TLV format. `Flags` is an optional AD structure that is not specific to *Apple* Continuity protocols, and that can be included by any devices to indicate their discoverable modes and capabilities.

byte each whereas `Value` is `Length`-byte long). In total, we have identified twelve different types of *Apple* Continuity messages that are listed in Table V.5.

Note that, for a given device, the *Apple* Continuity messages included in BLE advertisement packets can vary over time depending on its status and activity. For instance, advertisement packets generated by an iPhone will always include a Nearby Info message, and will occasionally [1] include a Handoff message.

V.2.3.1   Content protection

In practice, *Apple* Continuity messages are never encrypted as a whole. As part of the advertisement packets, their content is therefore exposed in clear. However, Proximity Pairing and Handoff messages include an encrypted payload (through AES-128 and 256 in ECB mode).

Similarly, other elements of data are not transmitted in clear but are hashed using SHA-256 then truncated. This is the case of Wi-Fi SSIDs, e-mail addresses and phone numbers found in AirDrop [206] and some Nearby Action messages (see Section V.7).

Finally, Handoff, Nearby Action and Nearby Info messages are authenticated through an `Auth Tag` that is computed using AES-256 in GCM mode [207].

---

1. When a compatible application is running.
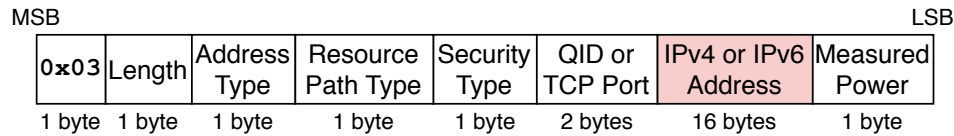
Figure V.2 – Format of the AirPrint message.

V.2.3.2 Use of random device addresses

In order to prevent physical tracking issues, we found that 1) advertisement packets including *Apple* Continuity messages are mostly using Random Non-resolvable or Random Resolvable addresses and 2) the Bluetooth Core Specification recommendation on the maximum lifetime of 15 minutes is enforced.

Nonetheless, we identified several exceptions. Indeed, AirPrint and AirPlay messages are using Public addresses, thus exposing the BD_ADDR. Analogously, AirDrop and "Hey Siri" messages use a Random Non-resolvable device address that never changes until the user turns off then on the Bluetooth interface of his device. As a consequence, such messages expose their users to tracking as their device addresses over which they are advertised remain stable over time.

V.2.4 Details of *Apple* Continuity protocols

In this section, we review the *Apple* Continuity messages found within BLE advertisement packets. For each message type, we present the associated service, the data format and the basics of the corresponding protocol (see Table V.5 for a summary of the *Apple* Continuity messages).

AirPrint: AirPrint is a feature included in *Apple* devices to discover compatible[1] printers and print documents via a wireless network without having to install additional drivers. Transmitted each time the user tries to print a document through AirPrint, those messages (see Figure V.2) include the complete IP address (IPv4 or IPv6) of the remote printer that can be leveraged as a first step toward a more elaborated attack [209].

AirDrop: AirDrop is a service which enables the transfer of files between *Apple* devices over Bluetooth and Wi-Fi. Transmitted each time the user attempts an AirDrop transfer, those messages (see Figure V.3) are composed of several 2-byte long fields containing hashed identifiers. Actually, we found that those hashes are the sixteen MSB of the SHA-256 digests of e-mail addresses and phone numbers configured into the iCloud accounts of the users (see Section V.7).

HomeKit: HomeKit is a framework for the monitoring and management of smarthome

---

1. This feature is built in most popular printer models, such as the ones listed on the *Apple* website [208].
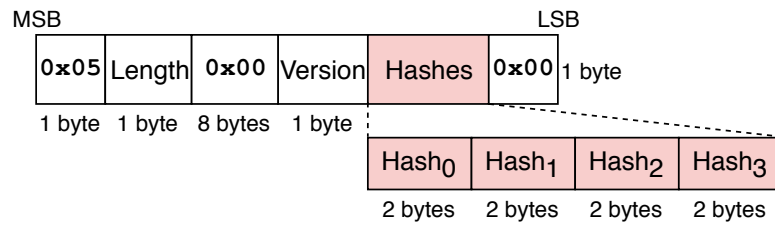
Figure V.3 – Format of the AirDrop message.
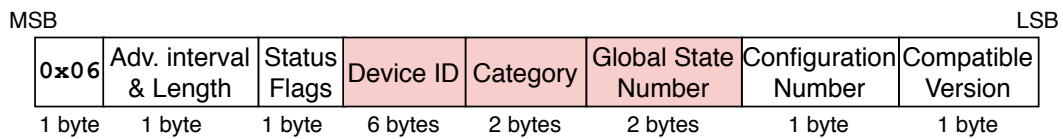
Figure V.4 – Format of the HomeKit message.

equipment supporting the HAP [202]. Note that, the HAP enables users to discover, configure and create actions to control HomeKit accessories. Constantly transmitted by powered HomeKit appliances, those messages (see Figure V.4) include fields identifying the device and its category, as well as a `Global State Number` (`GSN`) that is incremented each time the device state changes. In Section V.6, privacy implications of the `GSN` are further detailed.

Proximity Pairing: Proximity Pairing is a feature to facilitate the pairing procedure of audio devices with an iPhone or an iPad [210]. Constantly transmitted by active *Apple* audio devices (earphones and headsets), Proximity Pairing messages (see Figure V.5) include an encrypted payload concatenated to fields describing the device attributes (i.e. model and color), its position (i.e. in ear/case) through the `UTP` element and its current status: battery levels, charging status and lid open counter [1]. In Section V.3, tracking concerns inherent to those device status related fields are discussed.

"Hey Siri": Siri is the *Apple* virtual assistant that uses voice commands issued by a user to answer questions and perform actions. Broadcasted each time the user submits a command via voice activation, "Hey Siri" messages (see Figure V.6) include fields associated with the expressed Siri voice command: a `Perceptual Hash` (see Section V.8) and indicators of `Signal-to-Noise Ratio` (`SNR`) and `Confidence`. Likewise, those messages embed a field describing the `Device Class`. In Section V.8, we demonstrate how Siri voice commands can be inferred based on observations of the `Perceptual Hash`.

AirPlay: AirPlay allows wireless streaming of multimedia content between compatible devices. Transmitted each time the user tries to initiate the streaming of a content via AirPlay, those messages (see Figure V.7) are similar to the AirPrint ones as they include the complete IPv4 address of the AirPlay target in clear. In the context of network

---

1. A similar 1-byte lid close counter is suspected to be included too and used the same way the lid open counter is used.
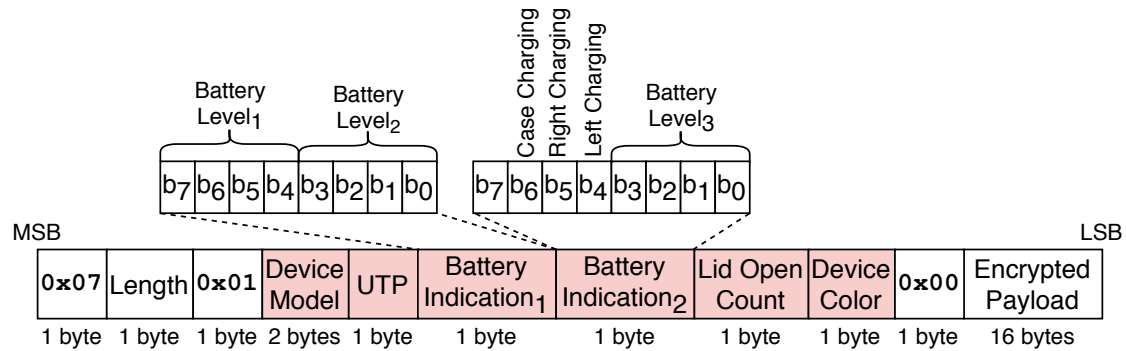
Figure V.5 – Format of the Proximity Pairing message.
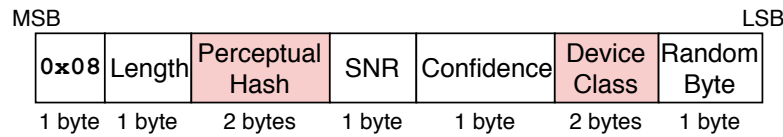
Figure V.6 – Format of the "Hey Siri" message.

intelligence gathering [211], we draw the attention to the fact that this conveyed IP address could be exploited by a remote attacker.

Magic Switch: The Magic Switch protocol is undocumented but seems to be related to the watchOS operating system. As observed by Martin et al. [1], only *Apple* Watches emit those messages. Transmitted when the Watch has lost the Bluetooth connection to its paired iPhone and when its screen is on, Magic Switch messages (see Figure V.8) include a 2-byte `Data` field that can expose the user to tracking (see Section V.3), followed by a `Confidence on Wrist` field that appears to be an indicator that the watch is worn (see Table G.1 in Appendix G).
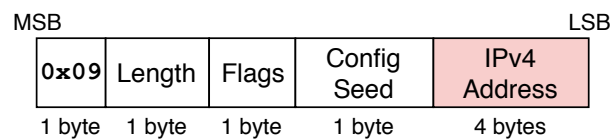
Figure V.7 – Format of the AirPlay message.
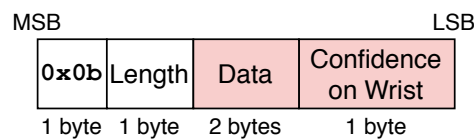
Figure V.8 – Format of the Magic Switch message.
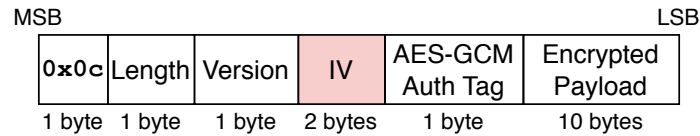
Figure V.9 – Format of the Handoff message.
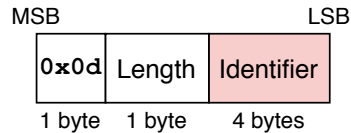
Figure V.10 – Format of the Tethering Target Presence (Instant Hotspot) message.

Handoff: Handoff [212] allows activities to be transferred between devices associated with the same iCloud account. As an illustration, a user who is browsing an article in Safari on its MacBook laptop can then move to its nearby iPhone (that is signed into the same iCloud account) and open the same webpage automatically on its smartphone. Transmitted each time the user interacts (i.e. opens, runs or closes) with a Handoff enabled application (e.g. Mail, Safari, Maps, Contact, etc.) [213], those messages (see Figure V.9) include an `Initialization Vector` (`IV`), a payload encrypted with AES-256 in ECB mode and an `Auth Tag` generated with AES-256 in GCM mode [207]. As reported by Martin et al. [1], we observed that the `IV` is not random and is incremented every time the payload changes (see Section III.3.4.2.1). Moreover, tracking issues raised by Handoff messages are presented in Section V.3.1 and Section V.4.1.

Instant Hotspot: Instant Hotspot is a feature designed to share the cellular connectivity of an iPhone or an iPad over Wi-Fi with nearby devices that are associated with the same iCloud account. De facto, this feature relies on two types of messages: Tethering Target Presence and Tethering Source Presence.

Transmitted each time the user does a Wi-Fi scan to discover surrounding Wi-Fi networks, Tethering Target Presence messages (see Figure V.10) include a 4-byte `Identifier` field that is generated from a `Destination Signaling Identifier` (`DSID`) tied to the iCloud account [214]. In practice, we witnessed that a scan is performed when the user opens or navigates in the Wi-Fi and network settings menu. Additionally, as observed by Martin et al. [1], the `DSID` is rotated only once every 24 hours involving tracking concerns (see Section V.3).

On the other hand, Tethering Source Presence messages are transmitted by *Apple* devices that are capable of sharing a cellular connection. Broadcasted in response to Tethering Target Presence messages coming from devices associated with the same iCloud account, those messages (see Figure V.11) include fields representing the current state of the device: `Battery Level`, nature of cellular connection (`Network Type`) and cellular `Signal Strength` (as reported by Martin et al. [1]).
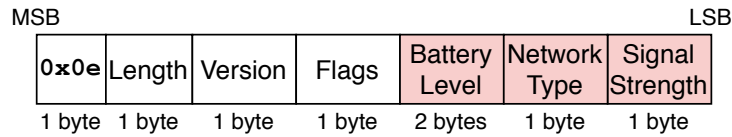
Figure V.11 – Format of the Tethering Source Presence (Instant Hotspot) message.

Nearby: Although it is not documented, the Nearby protocol appears to be used to inform nearby devices about the presence and state of a device. Furthermore, as they can include authentication tags, we assume that Nearby messages can be leveraged in authentication processes. In fact, this protocol uses two types of messages: Nearby Action and Nearby Info.

Transmitted each time the user takes an action[1] covered by the Nearby protocol, Nearby Action messages include a field describing the `Action Type` as well as a set of `Action Parameters`. In the cases of the iOS Setup (see Figure V.12) and Wi-Fi Password (see Figure V.13) actions, parameters contain characteristics of the device (i.e. class, model, color and OS version), but also hashed e-mail addresses and phone numbers of users configured into their iCloud accounts (see Section V.7).
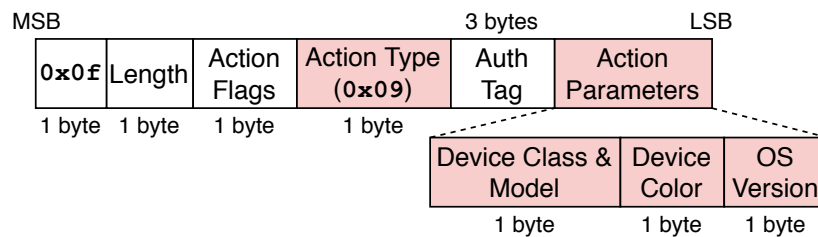
Figure V.12 – Format of the Nearby Action (iOS Setup) message.
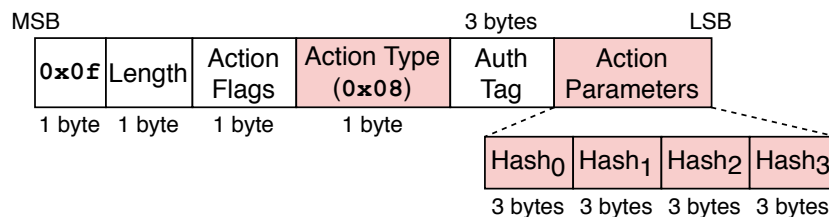
Figure V.13 – Format of the Nearby Action (Wi-Fi Password) message.

For their part, continuously transmitted by iPhones, iPads, MacBooks and Watches regardless of the device status (locked or active), Nearby Info messages (see Figure V.14) include an `Auth Tag` and an `Activity Level` field representing the current state of the device. In Section V.5, we present the activity inference menace associated with the

---

1. As an example, setting up a speaker, sharing a Wi-Fi password or answering a call (see Table G.2 in Appendix G for an extended list of Nearby Action types).
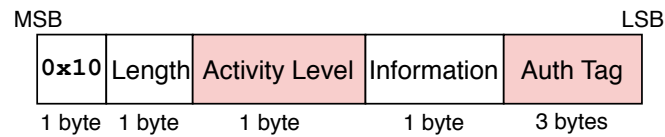
Figure V.14 – Format of the Nearby Info message.

`Activity Level` element, while the tracking threat related to the `Auth Tag` is exposed in Section V.3.

V.3   Passive tracking

In the context of wireless communications, the use of temporary and random link layer identifiers has been introduced as a countermeasure against physical tracking (see Section II.5). However, it has been shown that the content of frames can be used to fingerprint a device and link its distinct MAC addresses [77]. Applied to the BLE technology, this kind of issue has been also demonstrated [97] leveraging static identifiers and non-reset counters exposed within advertisement packets (see Section III.3.4).

In this section, we complement previous works by presenting a collection of new artifacts contained in *Apple* Continuity messages that can be leveraged for *passive* tracking purposes.

V.3.1   Identifiers and counters

Public addresses: AirPrint and AirPlay messages are transmitted by devices using their Public addresses. Similarly, we found that a MacBook laptop continuing an activity via Handoff broadcasts Nearby Info messages with its Public address.

Extended lifetime of Random Resolvable addresses: When a device is in power saving mode, it can keep the same Random Resolvable address for a duration that exceeds the Bluetooth Core Specification recommended maximum duration of 15 minutes [38, Vol 3, Part C, App. A]. Actually, we observed some Random Resolvable addresses that carry Nearby Info messages for more than four days.

Stable Random Non-resolvable addresses: Advertisement packets including AirDrop and "Hey Siri" messages use Random Non-resolvable addresses that are not changed over time. Indeed, we witnessed the same values for more than seven days.

Bad synchronization in pseudonym changes: As reported in [97] and [1], we attested that the change of the `Auth Tag` embedded in Nearby Info messages is not completely synchronized with the change of the device address: for a short duration after the address

change, the old `Auth Tag` is used with the new device address (see Section III.3.4.2.1).

Stable identifiers in AirDrop and Nearby Action (Wi-Fi Password) messages:
AirDrop and Nearby Action (Wi-Fi Password) messages include hashed e-mail addresses
and phone numbers that constitute stable identifiers [1].

Stable identifier in Tethering Target Presence messages: Tethering Target Presence messages contain a 4-byte long `Identifier` tied to the iCloud account that is rotated
only once every 24 hours, and thus acts as a stable identifier too.

Stable data in Magic Switch messages: The 2-byte long `Data` field found within
Magic Switch messages appears to be stable over time. In fact, we discovered that such a
field can remain unchanged during more than 30 minutes.

Non-reset `IV`: As part of the Handoff messages, the 2-byte long `IV` element is an
incremental counter that is not reset when the random device address changes, and
therefore that can be leveraged to link consecutive addresses (see Section III.3.4.2.1).

V.3.2   AirPods battery levels and `Lid Open Count`

Broadcasted by AirPods earphones, Proximity Pairing messages expose information about
battery levels and the `Lid Open Count` (see Section V.2.4) that can be exploited for
tracking [181]. Presented with a resolution of 10 levels, those messages include three
battery level indicators: one for each earphone, and one for the case. Also, the messages
contain a charging status, as well as a `Lid Open Count` field that is a 1-byte counter
incremented each time the case lid is opened. Note that, those information are only
emitted when the AirPods are outside the case, and the battery level of the case is solely
exposed when the lid is opened.

For our experiment, Proximity Pairing messages generated by AirPods have been recorded
during a day: the AirPods were intermittently used and put back in their case after each
usage session. Figure V.15 presents the evolution of the artifacts exposed by the benched
set of AirPods. Throughout this study, the battery level of the case decreases and the `Lid
Open Count` increases. Likewise, battery levels of earphones decrease while in use, and are
brought back to 100% by being stored in the case. Furthermore, it appears that battery
levels are quite identical for both earphones.

Alone, the `Lid Open Count` holds a potential for tracking. Indeed, it is a counter taking
16 values that can be used to identify the AirPods during a session of use (since it is
stable), and between two sessions (it is incremented by one). Concerning the battery levels
of the earphones, they take one of 10 values that evolves at a slower rate than the address

---

1. Also, as reported by Martin et al. [1], the `Hash₃` field of Nearby Action (Wi-Fi Password) messages
(see Figure V.13) contains the first three bytes of the SHA-256 hash of the SSID the client device is
attempting to join.
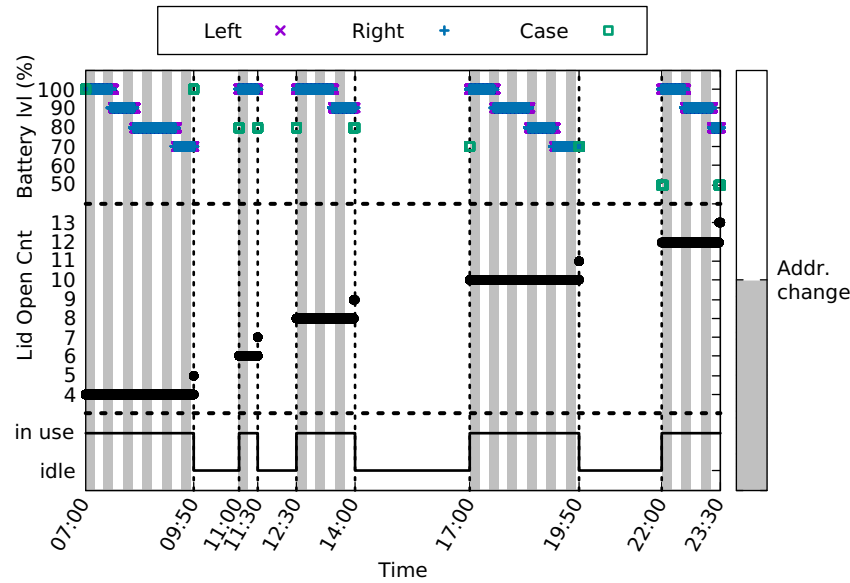
Figure V.15 – Evolution of battery levels and the `Lid Open Count` from Proximity Pairing messages broadcasted by a set of AirPods earphones. Different device addresses are represented by grey/white areas. Those data depict a day during which AirPods are used intermittently and the case is never charged. Battery levels and the `Lid Open Count` remain stable during periods longer than the lifetime of the random device addresses.

change. As a consequence, the combination of the `Lid Open Count` along with the battery levels of the AirPods can take a total of $16 \times 10 = 160$ different values.

In addition, we point out that the AirPods used during this study were brand new, and their battery levels were evolving almost exactly at the same rate. Nevertheless, we can imagine pairs of AirPods with a contrasted battery levels evolution. As a result, this would further improve the amount of identifying information.

Finally, we emphasize that the global energy of the system could be exploited as well to defeat the address randomization scheme. Actually, by modeling the energy transfer between the case and the AirPods, it would be possible to create a global energy fingerprint that could be leveraged to track a set of AirPods between usage sessions. In this regard, we leave this possibility for future work.

V.4   Active tracking/linking

As some *Apple* Continuity protocols are interactive, the reception of a message may trigger a reaction under the form of a new message emission. Leveraging this mechanism, eavesdroppers can then mount *active* tracking attacks in which they will replay messages to force a device to reveal its presence and identity.
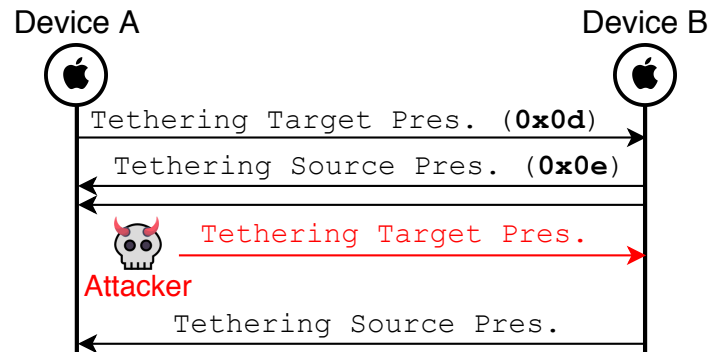
Figure V.16 – Replay of Tethering Target Presence (Instant Hotspot) messages as presented in [1]. By replaying such messages emitted by a *Device A*, an attacker triggers the emission of Tethering Source Presence (Instant Hotspot) messages by the *Device B* associated with the same iCloud account.

In most cases, a device will only react to messages coming from a known source: a device it has been previously paired with or that belongs to the same iCloud account. As a consequence, this feature implies that some messages will solely trigger a reaction from specific devices.

Introduced by Martin et al. [1], this kind of replay attack against an *Apple* Continuity protocol was illustrated through the replay of Tethering Target Presence (Instant Hotspot) messages that triggered a reaction from a source associated with the same iCloud account (see Figure V.16). Note that, Tethering Target Presence messages can be only replayed for at most 24 hours due to the rotating `DSID`.

In the following, we describe a novel active attack based on the replay of corrupted Handoff messages that allows an attacker to 1) track a device and its corresponding owner over time and 2) link devices associated with the same iCloud account.

V.4.1   Replay of corrupted Handoff messages

During our work, we discovered a new replay attack based on Handoff messages that are used to enable activity transfer between devices (see Section V.2.4).

From our experiments, we also found that Handoff messages are far more common than Tethering Target Presence ones and, in the Handoff protocol, source and destination roles can be endorsed by iPhones, iPads, MacBooks and Watches. Therefore, the Handoff replay attack can target a wider range of devices than the Instant Hotspot one.

In fact, this replay attack relies on resynchronization mechanism in which a device initiates a new synchronization procedure through a connection request when it receives a corrupted Handoff message: an authentication tag inconsistent with the payload and the `IV`, or an `IV`
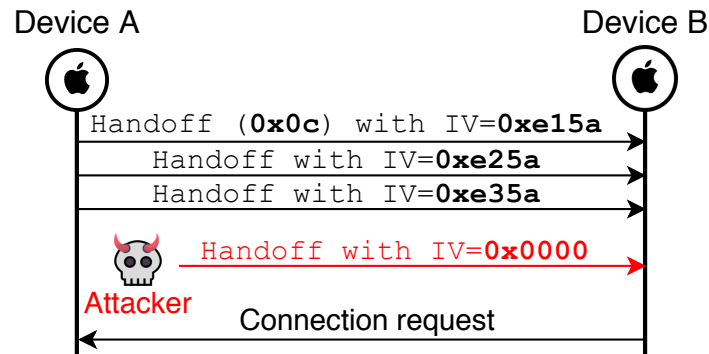
Figure V.17 – Replay of Handoff messages with a modified IV. By replaying a Handoff message including an IV less than the last one received (i.e. in this case, 0xe35a), the attacker forces *Device B* to initiate a new synchronization procedure with *Device A* through a connection request.

not greater than the last one received. Note that, in addition to the fact that we witnessed this behavior to be implemented in the `sharingd` binary, we confirmed it on our tested devices as well (see Table V.1).

To further illustrate the attack, Figure V.17 shows that such a resynchronization mechanism can be leveraged as follows:

1. the attacker captures a Handoff message emitted by a *Device A*;
2. the attacker later replays this message with a modified IV field (e.g. by setting it to zero);
3. a second *Device B* (associated with the same iCloud account) will respond to this message by initiating a connection, thus revealing its presence and its current device address.

By exploiting this attack, the eavesdropper is then able to detect the presence and the current address of any Handoff compatible device in range associated with the same iCloud account.

In principle, the presented Handoff based replay attack is similar to the Instant Hotspot one discovered by Martin et al. [1]. However, because it relies on Handoff messages, it is far easier to exploit.

Indeed, Handoff messages are broadcasted whenever the user interacts with one of the several compatible applications, whereas Tethering Target Presence messages are only advertised when a device searches for Wi-Fi connectivity (see Section V.2.4). Moreover, Handoff messages affect a much wider range of devices than Instant Hotspot ones: the source of Instant Hotspot messages is necessarily an iPhone, an iPad or a MacBook, and the destination can uniquely be a device with cellular connectivity (i.e. an iPhone or an iPad). To finish, Handoff messages can be emitted and received by all types of *Apple* devices, excluding AirPods.

Table V.1 – Experimental evaluation of replay attacks based on Handoff and Tethering Target Presence (Instant Hotspot) messages against different *Apple* devices.

Apple device	Model	OS version	Vulnerable to .... replay attack corrupted Handoff	Instant Hotspot
AirPods (2nd generation)	A1602	1A691		
iPad (5th generation)*	A1822	iOS 12.3.1	✓	
iPad Mini 3*	A1599	iOS 11.4	✓	
iPhone 6	A1586	iOS 11.4.1	✓	✓
iPhone 8	A1905	iOS 12.4	✓	✓
MacBook Air (13", 2014)	A1466	macOS 10.12.3	✓	
MacBook Pro (13", 2015)	A1502	macOS 10.13.6	✓	
Watch Series 2	A1757	watchOS 5.0.1	✓	
Watch Series 3	A1858	watchOS 5.1.3	✓	

* Only supports Wi-Fi (i.e. not *Cellular* capable).

V.4.2 Experimental evaluation of replay attacks

Leveraging a set of *Apple* devices, we appraised both the Handoff and Instant Hotspot [1] replay attacks.

To begin, each device was linked to an iCloud account. Thereafter, we captured a message (Handoff or Tethering Target Presence) emitted by a secondary device associated with the same iCloud account. In proximity of the benched device, we then replayed the message and monitored the wireless channel for a response. Finally, Table V.1 presents the results of the experimental evaluation where we can observe that the Handoff based attack affects all devices, except the AirPods. On the contrary, the Instant Hotspot replay attack only involves devices capable of sharing cellular connectivity such as iPhones and *Cellular* capable iPads [215].

V.4.3 Device linking

In addition to the tracking concerns, those described attacks can be leveraged to link devices associated with the same iCloud account, and thus belonging to the same user. Indeed, both Handoff and Instant Hotspot replay attacks trigger a reaction from other devices associated with the same iCloud account. In order to detect and identify other devices belonging to the same user, an attacker that has captured messages emitted by a particular device can then replay those messages later.

As each device potentially exposes different types of information, we point out that device linking increases the amount of information that can be gathered on an individual. Furthermore, such a privacy threat could be exploited to identify points of interest of the user by detecting devices that were left at home or at the office, for instance.

V.5 Exposed device status and characteristics

From our observations in Section V.2.4, some *Apple* Continuity messages include fields providing information on the status of a device (e.g. idle, screen on, playing video, etc.) or its characteristics:

Category and class are exposed by the `Category` field of HomeKit messages (see Table G.3 in Appendix G) as well as the `Device Class` of "Hey Siri" and Nearby Action messages (see Table G.4 and Table G.5 in Appendix G). Note that, analogous findings with regard to the disclosure of the device type (and model) through the `Manufacturer Specific Data` AD structure have been highlighted in Section IV.3.2.

Model is exposed by the `Device Model` fields of Proximity Pairing and Nearby Action messages (see Table G.6 and Table G.7 in Appendix G).

Color is exposed by the `Device Color` field of Proximity Pairing and Nearby Action messages (see Table G.8 and Table G.9 in Appendix G).

OS version is exposed by the `OS Version` field of Nearby Action messages (see Table G.10 in Appendix G).

In addition, the message type can be leveraged to fingerprint the device. As an illustration, Proximity Pairing messages are exclusively emitted by audio devices (earphones and headsets), while Magic Switch messages are only advertised by watches.

Similarly, elements describing the state of the device can be found in *Apple* Continuity messages:

Position (i.e. in ear/case) of the AirPods earphones is exposed by the `UTP` element of Proximity Pairing messages (see Table G.11 in Appendix G).

Activity of the device is exposed by the `Activity Level` field of Nearby Info messages (see Table G.12 in Appendix G) but also by the types of broadcasted continuity messages (e.g. AirPlay messages betray multimedia content streaming, AirPrint messages are involved in printing tasks, etc.).

State changes are exposed by the `Lid Open Count` and `Global State Number` field respectively contained within Proximity Pairing and HomeKit messages.

Cellular connectivity is exposed by both the `Network Type` (see Table G.13 in Appendix G) and `Signal Strength` fields of Tethering Source Presence (Instant Hotspot) messages.

Battery status is exposed through the `Battery Indication` and `Battery Level` elements of Proximity Pairing and Tethering Source Presence (Instant Hotspot) messages.

V.5.1   Applications

Inventory attacks: The information provided on the nature of the device can be used in *inventory attacks* [59] where the attacker will leverage the list of owned devices to infer attributes of a subject such as its wealth or medical condition. To the best of our knowledge, *Apple* Continuity protocols are not yet included in medical devices. Nonetheless, the ownership of *Apple* devices has been shown [62] to be a reliable indicator of wealth.

Visual identification: Physical characteristics of the device such as its category, class, model and color can be used for visual identification. In fact, it could serve to establish a link [68] between a visual identity (i.e. a person holding a device) and its radio identity (i.e. the device address embedded in BLE advertisement packets).

Event correlation: By exposing status changes in real-time, it is possible to correlate the identity of a device with specific events that occur. As an example, sending a message to an e-mail address will trigger a state change associated with a device address.

Activity monitoring: Exposure of a detailed description of device status has the potential to exhibit in real-time the activity of a user on its device, but also in its smart environment (see Section V.6). As such, this information could be then exploited to learn the habits of users, or to implement activity-based advertising systems [216].

V.6   Leaking smarthome activity

Broadcasted by powered appliances, HomeKit messages contain information that can be leveraged to infer the activity in a smarthome. Indeed, those messages include a GSN that is used to keep track of the device state changes [202, sec. 7.4.6.3], and that is incremented each time the state is modified.

For a number of smarthome devices, a state change is the consequence of an executed command (e.g. turning on/off a lightbulb) or a reaction to an environment modification induced by the user (e.g. the user passes by a motion sensor and its detection causes the increment of the GSN). Therefore, the state change in smarthome appliances can be an indicator of the presence and activity within a household [217].

To demonstrate this activity inference menace, we conducted an experiment in our laboratory involving two smarthome accessories that transmit HomeKit messages over BLE: a motion sensor (*Eve* Motion) and a connected lightbulb (*Osram* Smart+). Installed in our office, such advertised HomeKit messages were then recorded by our custom sniffer.

For each device, the GSN was extracted from the captured messages and translated into a temporal sequence of state changes. Figure V.18 presents the corresponding traces, in parallel with the time of presence and activity of the occupant.
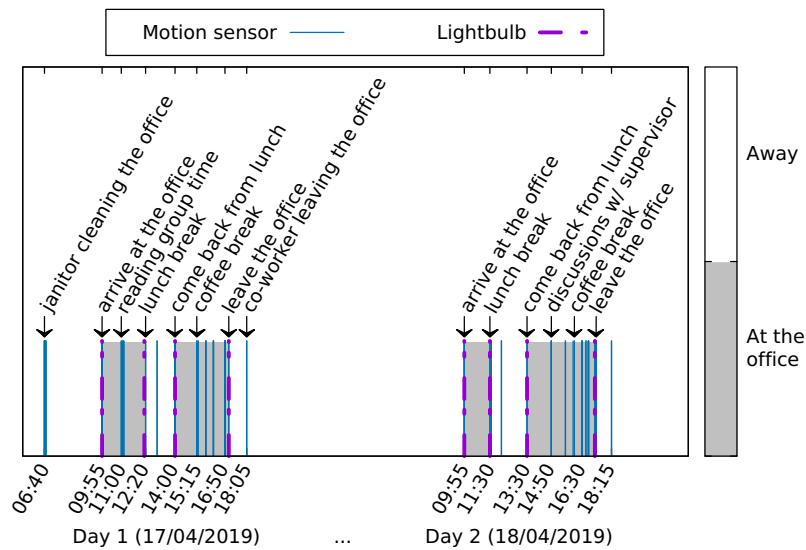
Figure V.18 – Representation of the GSN changes extracted from HomeKit messages during two days. For both the *Eve* Motion sensor and *Osram* Smart+ lightbulb, the GSN changes can be leveraged as an indicator that leaks the presence and activity of the user in its office.

In order to get an insight on the activity in the office, it is possible to exploit the GSN. First, any movement in the field of the motion sensor will induce a change of GSN, thus revealing the activity and presence in the room. Likewise, turning on or off the lightbulb will be reflected by a change of GSN. Note that, manually turning on/off the light using a physical switch leads to a GSN change as well.

As shown in Figure V.18, the GSN of the motion sensor is a good indicator of presence in the office, while the data provided by the lightbulb reveal the boundaries of the office work activity. Actually, it is important to mention that this attack does not allow the direct inference of a specific activity. Rather, it exposes a coarse grain information: in our case, moving in the room and using a light.

Lastly, HomeKit messages are broadcasted by various types of devices including door sensors, thermostats, and faucets (see Table G.3 in Appendix G). As a consequence, applying this attack to the whole range of HomeKit enabled devices has the potential to reveal detailed information on smarthome activities [217].

V.7 Leaked e-mail addresses and phone numbers

During our reverse engineering of *Apple* Continuity protocols (see Section V.2.4), we observed that AirDrop and Nearby Action messages can include hashed e-mail addresses, phone numbers and SSIDs. In the following, although those identifiers are not exposed in clear, we describe how their values can be recovered through *guesswork* attacks [218, 219].
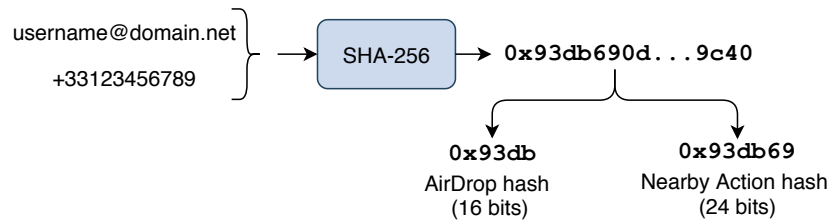
Figure V.19 – Description of the process used to compute hashed identifiers found within AirDrop and Nearby Action messages. The identifier (i.e. an e-mail address or a phone number) is first hashed with SHA-256 then truncated to keep only the 16 and 24 MSB respectively for AirDrop and Nearby Action messages.

By analyzing the `sharingd` binary, we found that the identifiers are hashed using the SHA-256 function (without any salt), and are then truncated to 16 or 24 bits respectively for AirDrop and Nearby Action messages (see Figure V.19).

Through experiments with a device linked to an iCloud account on which two e-mail addresses and one phone number were configured, we confirmed the aforementioned information. Indeed, after triggering the emission of AirDrop and Nearby Action messages, corresponding hashes were extracted from captured messages, and were matching with the 16 (respectively 24) MSB of the SHA-256 digests of the identifiers.

V.7.1   Recovering hashed identifiers

As a cryptographic hash function, SHA-256 is resistant to pre-image attacks. In other words, given a digest d, finding x such that SHA-256$(x) = d$ is computationally infeasible. Nonetheless, when the set from which the values are drawn is small enough, re-identification through a brute force attack may become practical. Also called *guesswork*, the brute force attack for re-identification has been successfully employed against hashed e-mails and other digital identifiers [218, 219, 220].

At first, let us consider a hash function $h()$ that produces digests of length l and a digest d that has been obtained by hashing the value $\bar{x}$, i.e. $h(\bar{x}) = d$. In fact, the objective of the guesswork is to find all the pre-images of d in a set S of possible values. As a result, a guesswork attack will proceed by hashing all the elements in S and returning any element $s \in S$ such that $h(s) = d$. In our case, $h() = \text{MSB}_k(\text{SHA-256}())$, where $\text{MSB}_k()$ represents the function returning the k MSB of the input.

In term of time, the cost of the guesswork can be expressed as: $c = m/z$, where m is the size of S (i.e. the number of elements to be tested) and z the hashing speed of the attacker.

Also, another parameter to consider is the potential false positives returned by the

guesswork: value x such as $h(x) = d$ but $x \neq \bar{x}$. Indeed, m, the size of the set S, is potentially larger than the number of hash values n. As a consequence, it is likely that a given hash can have several pre-images in the set S. Described in [219], this case where $m > n$ falls under the *many-to-one* model, and the average number of elements of S matching a given digest is $r = m/n$.

In the general case, and assuming that the identifier used to produce the hash is included in the set (i.e. $\bar{x} \in S$), the average number of identifiers returned by the guesswork [219] is the following:

$$r = \begin{cases} m/n & \text{if } m > n \\ 1 & \text{otherwise} \end{cases} \tag{V.1}$$

Since the attacker is only interested in the value $\bar{x}$ (i.e. the identifier associated with the iCloud account), the number r should be kept as small as possible.

V.7.1.1   A posteriori confirmation

In the case where the guesswork returns several identifiers, the attacker will need additional information to narrow down the one used by the iCloud account.

To perform this confirmation, one possible approach is to exploit the device status information leaked by Nearby Info messages through an *event correlation* attack (see Section V.5.1). Actually, the attacker can send messages to each identifier while monitoring for a device that goes from idle (code `0x03`) to active (code `0x07`) (see Table G.12 in Appendix G). In the same way, the attacker can rely on visual observations: the target will pick up its phone or read the message.

V.7.2   Identifier sets for the guesswork

To be tested against a hashed value, the guesswork attack requires a set of elements S. As previously mentioned, the size of this set will have an impact on efficiency of the guesswork (i.e. on the cost and number of false positives). Therefore, the set S should be as small as possible.

In this section, we discuss the potential approaches available to the attacker in order to build such a set of identifiers.

V.7.2.1   E-mail addresses

Basically, e-mail addresses can be up to 307 characters[1] long [219] corresponding to 2^{1666} different values. As a result, it is not realistic to envision an exhaustive search over all the possible values, and the attacker will have to rely on dictionary of practical size for the attack.

To build such a dictionary, an attacker can either obtain a list of existing e-mails or try to generate e-mail addresses by assembling names and domains. For the sake of simplicity, we will focus on the first case (i.e. list of existing e-mails), and refer the interested readers to previous works done on the synthesis of e-mail addresses and identifiers [219].

To evaluate our guesswork attack, we then considered several lists that can be used by an attacker:

— all existing e-mail addresses worldwide;
— e-mails found in leaked database [221];
— e-mails of Gmail users [222];
— e-mails of our university department;
— e-mails of PETS 2019 participants [223].

V.7.2.2   Phone numbers

In practice, phone numbers follow a general format that represents 8.11×10^{14} possible values [220]. Nevertheless, this space can be significantly reduced if the attacker has additional information on its target such as the region in which its phone number is registered. Moreover, this set can be further truncated by selecting a subdivision of the geographical regions or number ranges of specific operators. In particular, the attacker can focus on phone numbers of mobile operators as iCloud accounts are usually configured with a mobile phone number. Note that, a resourceful attacker can also have access to a database of all registered phone numbers in a region.

In order to evaluate a potential attack, we considered several sets that an attacker can use:

— all possible values following the number format of a region;
— all possible values following the mobile number format of a region;
— all registered mobile/landline numbers in the region.

For our study, we selected *France* as region.

---

1. 54 and 253 characters respectively for the username and the domain.

Table V.2 – Evaluation of hypothetical guesswork attacks on hashed e-mail addresses and phone numbers. For each set of identifiers, the corresponding size is used to evaluate the cost of the attack and the average number of values that will be returned as matching the digest (supposing the set includes the original identifier). The hashing speed of the attacker is assumed to be 2000 kH/s.

	Set size	Guesswork time (s)	Average number of matching identifiers	
			16 bits (AirDrop)	24 bits (Nearby Action)
E-mail addr. All existing addresses	4.7×10^9	3600	71716	280
Leaked database [221]	773×10^6	387	11795	46
Gmail users [222]	1.5×10^9	750	22888	89
University department	7000	< 1	1	1
PETS 2019 participants [223]	247	< 1	1	1
Phone nb. France	10^9	500	15259	60
France, mobile	2×10^8	100	3052	12
France, registered mobile [225]	77.2×10^6	39	1178	5
France, registered landline [225]	37.8×10^6	19	577	2

V.7.3 Simulation results

Leveraging different sets of e-mail addresses and phone numbers, Table V.2 presents the outcome of the hypothetical attacks. For our simulation, it is important to mention that we assumed the attacker to be able to test identifiers at the rate of 2000 kH/s [1].

As the worst-case scenario (i.e. all the existing e-mail addresses) can be tested within an hour (3600 seconds), the attack cost appears to be reasonable and practical in every cases.

Ranging from one up to several thousands, the number of matching values depends on the size of the digest. For a 16-bit long digest of AirDrop, the list of values returned by the attack can contain several thousand identifiers. However, with the 24 bits of the Nearby Action, the set is much smaller and can be, in some cases, curtailed to a single element. Finally, the hashed identifier can be recovered without ambiguity in a matter of seconds when the set of potential value is small enough (i.e. several thousand identifiers).

V.8 Voice Assistant commands

When receiving voice commands, a Siri enabled device will emit "Hey Siri" messages including a digest of the command under the form of a `Perceptual Hash` (see Section V.2.4).

In this section, we analyze how this data can be leveraged to gain knowledge on spoken commands.

---

1. This hashing speed has been observed using `hashcat` (v5.1.0-951-g6caa786) [224] in straight (dictionary) attack mode running on an Intel® Core™ i7-5600U CPU @ 2.60 GHz.

V.8.1   Perceptual hashing

Commonly used for content identification [226], perceptual hashing is a technique to compute the digest of a signal such as an image or a sound. In fact, a key property of the perceptual hashing is that two perceptually similar signals $x \sim x'$ should produce digests at a close distance, $d(h(x), h(x')) < \varepsilon$, where ε is a threshold representing the robustness of the hashing algorithm [227].

For audio signals, one state of the art technique is the *Philips Robust Hash* (PRH) [226]. Producing a 32-bit long digest of an audio signal, this algorithm first decomposes the signal in overlapping frames prior to apply the following process to each frame: switching to the frequency domain using a Fast Fourier Transform (FFT), decomposition of the signal into 33 non-overlapping bands and computation of the energy in each band. Then, it generates a 32-bit digest in which the bits depend on the energy variation between consecutive bands and frames. For a detailed presentation of the algorithm, we point the interested readers to the work of Haitsma and Kalker [226].

V.8.2   Observations on Siri's perceptual hash

Twice as short as the output of the standard PRH algorithm, the `Perceptual Hash` included in "Hey Siri" messages is 16-bit long. Moreover, by analyzing the `CoreSpeech` binary, we identified that the `pHash()` function is in charge of computing this perceptual hash.

Actually, we were unable to fully reverse engineer such a function. Nonetheless, we were capable of identifying several characteristics: it takes an audio signal as input, it returns a 16-bit integer, it uses a FFT, it computes energy in the frequency domain, and bits of the hashes are computed based on the differences between consecutive elements of a buffer. Together, all those features suggest that this code implements a variant of the PRH algorithm.

To continue our study, we analyzed the stability of the perceptual hashes among different users by setting up an experiment that involved two male individuals. Table V.3 presents the experimental results where we can notice that the same commands produce significantly different hashes when spoken by two different users. As a consequence, this indicates that the perceptual hash depends on the command but also on the user: the speaking speed, the voice tone as well as the pronunciation appears to have an impact on the generated digest.

Table V.3 – Perceptual hashes obtained from voice commands issued by two male individuals. The Hamming distances are computed between hashes of $User_A$ and $User_B$ considered as binary values (i.e. sequences of bits).

Voice command	Perceptual Hash		Hamming distance
	$User_A$	$User_B$	
Call Mark.	0xeff0	0xb608	9
Call Bob.	0xfdf0	0xaf08	8
Play some music.	0x9f82	0x9547	6
Search the web for 'privacy'.	0x39ef	0xfe81	10
Send a message to Mark.	0x1438	0xb3b4	8
Send a message to Bob.	0x10b0	0xb31c	8
Set a timer for 3 minutes.	0xb680	0x181f	11

V.8.3 Exploiting Siri's perceptual hash

As noticed by Knospe [227], the compactness of the digest prevents the reconstruction of the original audio signal. However, because it is a robust and identifying representation of an audio signal, the digest can still reveal information about the voice command.

In this section, we show how perceptual hashes can be leveraged by a *passive* attacker to infer commands spoken by a remote user.

V.8.3.1 Dictionary attack on perceptual hashes

By definition, the perceptual hashing implies that a given command should produce the same output modulo small variations. Knowing the digest of a given command, it would be thus possible for an attacker to infer the command issued to the voice assistant.

For this experiment, we supposed that the attacker has captured a perceptual hash $\bar{y}$ that has been produced when the target spoke the command $\bar{x}$ to the voice assistant, i.e. $\bar{y} = h(\bar{x})$. Based on the captured hash $\bar{y}$, the aim of the attacker is then to identify the command $\bar{x}$. Also, we further assumed that the attacker has a dictionary that contains a list of commands and their corresponding digests: $\mathcal{D} = \{(x_i, y_i)\}_{0 \leq i < n}$, where y_i is the digest of the command x_i, and n the size of the dictionary.

Using the dictionary $\mathcal{D}$, the attacker can perform a custom dictionary attack by finding the command $x_k \in \mathcal{D}$ such that $y_k = h(x_k)$, and y_k is close enough to $\bar{y}$: $k = \text{Argmin}(d(y_i, \bar{y}))_{0 \leq i < n}$ and $d(y_k, \bar{y}) \leq \varepsilon$. Algorithm 2 describes the procedure of this attack that takes the intercepted hash and the dictionary as inputs, and returns a command of the dictionary or nothing if it has not found a close enough hash.

Input: $\mathcal{D} = \{(x_i, y_i)\}_{0 \leq i < n}, \bar{y}, \varepsilon$
Output: x' such that $y' = \mathcal{D}(x')$ and $y' = \mathrm{Argmin}(d(y, \bar{y}))_{y \in \mathcal{D}}$ and $d(y', \bar{y}) < \varepsilon$
$x' = \varnothing$;
$r = +\infty$;
for $(x, y) \in \mathcal{D}$ **do**
    if $d(y, \bar{y}) <= \varepsilon$ **then**
        if $d(y, \bar{y}) < r$ **then**
            $x' = x$;
            $r = d(y, \bar{y})$;
        end
    end
end

Algorithm 2: Dictionary attack on perceptual hashes.

V.8.3.2 Building a dictionary of commands and digests

To be practical, the attack requires that the attacker holds a dictionary of commands and hashes. Prior to the attack, such a dictionary could be created during a learning phase in which the attacker would be able to capture voice commands and perceptual hashes: as an example, by being physically present in the same room. Likewise, another approach could be to monitor the BLE advertising traffic and correlate perceptual hashes with observable events such as network traffic and state changes of smarthome appliances (see Section V.6).

To build our dictionary, we considered the possibility of using a voice synthesizer. Nevertheless, as observed in Section V.8.2, the values of the perceptual hashes depend on the speaker. Therefore, it is not possible to build a universal dictionary. Note that, this issue could be tackled with the help of machine learning to mimic the voice of the target [228]. In this regard, we leave this direction for future work.

V.8.3.3 Evaluation

In order to evaluate the performance of the dictionary attack, we leveraged a dictionary A of hundred commands (see Table H.1 in Appendix H) that was built based on a list of common Siri commands [229]. Then, each command was spoken to the device and the corresponding perceptual hash was recorded.

Afterwards, we played a second set B of commands composed of two sets: B_1 including fifty commands from the dictionary A, and B_2 including fifty commands external to the dictionary A (see Table H.1 in Appendix H). Based on the corresponding hashes, we ran the dictionary attack for various values of the threshold ε.
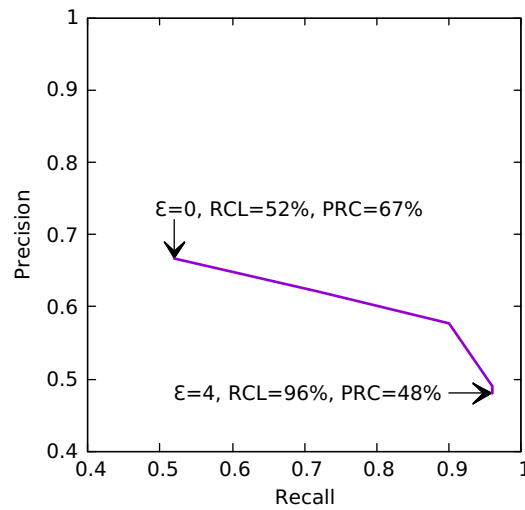
Figure V.20 – Performance evaluation of the dictionary attack on Siri's perceptual hashes. The precision and recall are presented as a parametric curve computed using the threshold ε. The distance threshold ε covers the interval $[0; 4]$.

To measure how successful the attack was, we rely on its precision and recall [230]. In our case, the precision is defined as the proportion of correctly matched commands among all matched commands, while the recall is characterized as the proportion of correctly matched commands among B_1. Figure V.20 presents the precision and recall of the dictionary attack.

For a threshold of $\varepsilon = 0$ (i.e. an exact match of the perceptual hashes), a precision as high as 67% with a recall of 52% can be obtained. By taking larger values of ε, this will relax this constraint leading to higher recall at the cost of a precision reduction.

In addition, due to the small size of our dictionary, the performance presented above is likely to be overestimated. Indeed, if the size of the dictionary increases, the distance between hashes will diminish and increase the probability of collision. Besides, there is an upper bound [1] for the number of distinguishable commands that can be expressed as $2^{16-2\varepsilon}$. As a result, for $\varepsilon = 2$, the number of distinguishable commands is at most $2^{12} = 4096$.

V.9 Impact of the attacks

In this section, we examine the impact of the presented attacks on the privacy of users.

To start, we consider the feasibility of those attacks with regard to the range. Although BLE was designed to provide a range of up to a hundred meters in outdoor environments [231],

---

1. Perceptual hashes can be seen as words of a code of length 16 with correction capacity ε. Thus, the number of distinguishable hashes is the dimension of the code k. From the Singleton bound and the definition of minimal distance: $k \leq 16 - 2\varepsilon$.

Table V.4 – Experimental evaluation of the range of *Apple* Continuity BLE messages leveraging various *Apple* devices in different environment settings.

Apple device	Model	Range (in meters)				
		Indoor (#walls*)				Outdoor
		0	1	2	3	
AirPods (2nd generation)	A1602	49	45	41	39	61
iPad (5th generation)	A1822	51	49	44	41	63
iPad Mini 3	A1599	50	48	44	42	62
iPhone 6	A1586	50	46	43	38	62
iPhone 8	A1905	51	49	46	42	65
MacBook Air (13", 2014)	A1466	52	50	47	45	65
MacBook Pro (13", 2015)	A1502	52	51	49	47	67
Watch Series 2	A1757	50	47	45	42	61
Watch Series 3	A1858	51	49	46	43	62

* Walls are constituted of plasterboard and are about 15 centimeters thick.

we witnessed that it is usually shorter in practice. Indeed, we measured the BLE range of several advertising *Apple* devices and found that they can be all received at least to 61 and 38 meters respectively in outdoor and indoor environments (see Table V.4). As such, this means that our attacks can be executed from a significant distance like from the other side of the street or from another room, for instance.

As some messages may be only available in certain conditions, a second element to consider is then the context in which the attack can be achieved. As an illustration, Nearby Info messages are continuously available thus constantly exposing the user to the corresponding tracking threat. On the other hand, AirDrop messages, that can be harvested for identifiers (see Section V.7), are only available upon a file transfer. Table V.5 summarizes the context in which the messages are broadcasted along with their corresponding threats.

Provided that the aforementioned conditions are met, most of those attacks are straightforward to implement. However, hashed identifiers and voice commands recoveries may be more complex to perform: they both require prior knowledge under the form of a dictionary, and the voice command attack can be negatively impacted by speech variations (see Section V.8.2).

To finish, a last aspect to consider is the privacy risk [232] associated with the elements of information obtained by a potential attacker. Through physical tracking and activity inference, users may be affected by stalking, surveillance and burglary. More worrisome, e-mail addresses and phone numbers may expose users to spear-phishing and malicious account recoveries. Overall, we highlight that most information elements could be leveraged to profile the user as they reveal identifiers, activities, whereabouts and owned devices.

Table V.5 – Summary of the *Apple* Continuity messages (CM) with their conditions of emission and associated privacy threats.

Apple CM type	Code	Message emission	Privacy threats			
			Activity infer.	Device attr.	Tracking	User info
AirPrint	0x03	On user action	✓		✓	
AirDrop	0x05	On user action	✓		✓	✓
HomeKit	0x06	Constantly	✓	✓		
Proximity Pairing	0x07	Constantly		✓	✓	
"Hey Siri"	0x08	On user action		✓		
AirPlay	0x09	On user action	✓		✓	
Magic Switch	0x0b	On user physical* action		✓	✓	
Handoff	0x0c	On user action			✓	
Instant Hotspot						
Teth. Target Pres.	0x0d	On user action	✓		✓	
Teth. Source Pres.	0x0e	Reaction to *Target Pres.*		✓	✓	
Nearby						
Nearby Action	0x0f	On user action	✓	✓	✓	✓
Nearby Info	0x10	Constantly	✓		✓	

[*] The *Apple* Watch loses the Bluetooth connection to its paired iPhone and its screen is on.

V.10   Recommendations

In this section, we present a set of recommendations for the design and implementation of continuity protocols. Complementing the ones introduced in [97] and [1], we point out that all the following protection measures should be addressed by *Apple*.

Encryption and content minimization of continuity messages: When possible – e.g. when devices are associated with the same iCloud account and share cryptographic keys, or when such keys are exchanged during a pairing procedure between two devices – the content of *Apple* Continuity messages should be encrypted [1]. In case the data cannot be encrypted, the content exposed in clear should be kept to a bare minimum: for instance, by avoiding the exposure of the device status and characteristics (see Section V.5).

Timestamps: Timestamping *Apple* Continuity messages is a simple countermeasure that could be considered to defeat the replay attacks detailed in Section V.4. Furthermore, most *Apple* devices have a local clock and broadcasted messages already contain authentication tags. Therefore, including a timestamp with a coarse granularity (i.e. seconds or minutes) would be enough to prevent long term tracking while avoiding to expose users to clock-based fingerprinting [233].

Synchronization between continuity protocols and device address changes: Identifiers, counters and data embedded in continuity messages can expose users to tracking if they are not rotated exactly at the same time as the BLE device address (see

---

1. This is the approach followed by *Microsoft* within the CDP protocol [148, sec. 2.2.2.2.3].

Section V.3). To tackle this privacy menace, the rotation of the address and the content of *Apple* Continuity messages should be then carefully synchronized. Note that, the BLE device address and continuity protocols do not belong to the same network layer. Thus, from a technical point of view, this could make the synchronization challenging.

Use of temporary random device addresses when advertising: At the exception of smarthome appliances, most *Apple* devices are made to be carried by users over time. To undermine related physical tracking issues, continuity messages that are advertised over a stable device address (see Section V.2.3.2) have to implement the privacy provisions of the LE Privacy feature [38, Vol 3, Part C, sec. 10.7]. Moreover, in order not to be predictable by a remote attacker [1], the rotation of the temporary pseudonyms should be done stochastically instead of relying on a 15 minutes fixed timer.

V.11   Conclusion

To conclude, this chapter disclosed a collection of privacy issues in the *Apple* Continuity protocols that range from mild leakages, such as the exhibition of device models, to serious leakages such as the exposure of PII (i.e. e-mail addresses and phone numbers).

First, we defined the ins and outs of continuity protocols before to present the methodology that we used to reverse engineer [1] the suite of *Apple* Continuity protocols. General features of *Apple* Continuity messages are discussed as well and we witnessed that, even if *Apple* devices have cryptological capabilities, such messages are never encrypted as a whole. Besides, we pointed out that messages of AirPrint, AirDrop, "Hey Siri" and AirPlay are advertised over Public and stable Random Non-resolvable device addresses, trivially exposing their users to physical tracking.

Second, we made an inventory of identifiers and counters carried by *Apple* Continuity messages that can be leveraged for *passive* tracking purposes. Beyond those artifacts, we also demonstrated that the `Lid Open Count` and battery levels of AirPods earphones embedded within Proximity Pairing messages hold together a potential for tracking users. Moreover, we highlighted that fingerprinting the system through its global energy can constitute an additional privacy threat that would further improve the amount of identifying information.

Third, we discovered that the interactivity of *Apple* Continuity protocols can be leveraged to mount *active* tracking attacks for which the principle is simple: by replaying captured messages, the attacker will force a device to reveal its presence and identity. To follow this finding, we described then experimentally evaluated a novel active attack based on the replay of corrupted Handoff messages. In addition to tracking, we emphasized that such a replay attack can serve to link *Apple* devices belonging to the same user too.

---

1. As the detailed research required an in-depth understanding of *Apple* Continuity protocols for which most specifications are not public, we underline that this work of reverse engineering is a key point of our study.

Fourth, we compiled an exhaustive list of device status and characteristics broadcasted in clear within *Apple* Continuity messages. From the device class to its battery status, through its model and color, we showcased that such fields provide meaningful information on the device that could lead to a plethora of privacy invasive applications. In this regard, we detailed four of those applications, namely inventory attacks, visual identification, event correlation and activity monitoring.

Fifth, we found that smarthome appliances, such as an *Eve* motion sensor and an *Osram* connected lightbulb, include a `Global State Number` (`GSN`) in their HomeKit emitted messages that a *passive* attacker can leverage to gain knowledge on the presence and activity of an individual within its household. More specifically, we showed that such a `GSN` is each time incremented in reaction to an environment modification induced by the user, or when this latter triggers a particular command.

Sixth, we demonstrated how hashed e-mail addresses and phone numbers exposed within AirDrop and Nearby Action messages can be recovered through *guesswork* attacks. To this end, we selected different sets of identifiers and provided insights on how to build a dictionary of practical size prior to simulate those attacks. Afterwards, we analyzed the experimental results, and asserted that recovering hashed identifiers without ambiguity is possible in a matter of seconds when the set of potential value contains several thousand identifiers.

Seventh, we detailed how the perceptual hash advertised over "Hey Siri" messages can be passively exploited in order to infer spoken commands of a remote user. To start this study, we introduced the *Philips Robust Hash* algorithm that is one state of the art perceptual hashing technique. Thereafter, we followed our work by analyzing the `CoreSpeech` binary in order to understand how *Apple* devices compute such a hash. Ultimately, we presented how to build a dictionary of hundred commands and digests before to experiment a dictionary attack on perceptual hashes of Siri. From our results, we observed that a precision as high as 67% with a recall of 52% can be obtained for an exact match of the perceptual hashes.

Last but not least, we evaluated the impact of the reported attacks relying on several elements, namely the feasibility of such attacks with regard to the range, the context in which they can be performed, and the privacy risk resulting from the information that a potential attacker can obtain. Furthermore, upon some minimalist conditions are met (i.e. the attacker is in range and is able to capture the *Apple* Continuity messages broadcasted by its target), we attested that most of those attacks are straightforward to implement.

Finally, to protect users against the uncovered privacy threats, we provided a set of recommendations that should be taken into account during the design and implementation of *Apple* Continuity protocols. From the encryption and content minimization of continuity messages to the use of temporary random device addresses when advertising, we pointed out that all the described countermeasures should be addressed by *Apple*.

V.12   Concluding remarks

Several severe security and privacy threats have been exposed in *Google* [234] and *Apple* [1, 235] continuity protocols. In all cases, the specifications were not public and the authors had to rely on reverse engineering to understand the system prior to identify its flaws.

As a result, this is yet another demonstration that *security through obscurity* does not work. Also, this shows that even companies with extended resources and dedicated security/privacy teams cannot only rely on internal scrutiny of their systems to avoid such issues.

To specify dedicated protocols, or at least to set up guidelines for the design and implementation, we believe that those technologies could benefit a joint standardization effort [1] with security researchers.

To finish, it is important to mention that the privacy issues demonstrated in this chapter were reported to *Apple*, *Eve* and *Osram* on May 29th, 2019. However, as of July 7th, 2020, none [2] of those issues have been corrected. Given the popularity of the appliances produced by *Apple* and partner companies, hundreds of millions users are then currently impacted as the discovered concerns apply to all the 1.5 billion active *Apple* devices worldwide [199]. To make the matter worst, most of the presented vulnerabilities can be *passively* exploited by a remote eavesdropper.

Note that, in order to allow the reproduction of our research results, we publicly released a `Wireshark` dissector [22] that parses messages advertised by the BLE *Apple* Continuity protocols.

---

1. Since continuity protocols are transversal to several wireless technologies, such standardization works could take place at IETF rather than in technology-specific organizations such as IEEE 802 and Bluetooth SIG.

2. At the exception of the exposed `Lid Open Count` that is now reset each time the AirPods are back in case.

Chapter VI

Conclusion

This chapter provides the final conclusion with regard to the work done on privacy issues induced by the use of wireless communicating IoT devices. At the beginning, a summary relating our studies on the physical tracking and inference of users information threats is given in Section VI.1. Then, Section VI.2 outlines the limitations of this thesis as well as its perspectives for future work. The industrial, social and media impacts that stem from our research are detailed in Section VI.3. At the end, Section VI.4 furnishes concluding remarks highlighting that the users privacy cannot be fully achieved through the sole technical side.

VI.1 Summary

There is a clear momentum in the development of the IoT, and a growing pressure for more privacy from both regulators and consumers. During this thesis, we then aimed at strengthening the research activity on the IoT, and especially on its radio link. As a result, we presented privacy challenges that wireless communications of the IoT devices can raise.

State of the Art: By introducing service discovery mechanisms in wireless devices, we began our state of the art. Afterwards, we demonstrated that radio signals are a source of privacy leaks that can fall into two categories: *physical tracking* and *inference of users information*. In the context of wireless technologies, we reviewed existing privacy considerations. As of today, we found that device address randomization as well as data encryption are the only mechanisms used by manufacturers to preserve the privacy of their customers. As a reminder, device address randomization is intended to replace the link layer identifier with a temporary and random one in order to tackle physical tracking issues, while data encryption is the process of encoding information to prevent unauthorized access. However, we pointed out that such mechanisms alone cannot protect the privacy of users. More worrisome, even if data are encrypted, metadata can also betray sensitive information.

After this survey, we gave an overview of the IoT deployment prior to show that its massive adoption increases wireless communications proportionally to its related privacy threats. To finish our state of the art, we point out that privacy protection regulations and entities such as the GDPR, the ePR and the CNIL conduct efforts to slow down the deployment of wireless based analytics systems.

Tracking users leveraging advertising data and metadata: Physical tracking was the first privacy threat that we dealt with in this thesis. To study this menace, we leveraged two datasets[1] crowded with BLE advertisement packets ($dataset_{Passive}$) and GATT profiles ($dataset_{Active}$) that we collected in the wild. To minimize the privacy risks associated with the captured data, we further presented how we anonymized and sanitized such datasets before to describe *passive* and *active* attacks. As its name suggests, passive attacks are the most devious as the eavesdroppers leave no traces on wireless channels.

To start the first milestone of this thesis, we made some observations on $dataset_{Passive}$. Although the LE Privacy feature appears to be widely adopted, we discovered that some manufacturers still use Stable devices addresses when advertising, trivially exposing their users to tracking. Note that, the LE Privacy feature was introduced in Bluetooth Core Specification version 4.0 [39, Vol 3, Part C, sec. 10.7] and specifies the use of temporary random pseudonyms. Despite those provisions, we analyzed the device address randomization and uncovered that some random addresses have lifetimes exceeding the recommended maximum duration of the Bluetooth Core Specification, while some others seem not to be randomly generated.

Based on our *passive* attacker model whose objective was to defeat the device address randomization scheme leveraging advertising data and metadata, we exposed that static identifiers and non-reset counters included in advertisement packets of manufacturers such as *Apple*, *Microsoft* and *Google* can hamper the anti-tracking feature.

In order to raise public awareness toward physical tracking concerns that can stem from wireless technologies, we implemented *Venom*, a *Visual and ExperimeNtal BluetOoth Low Energy tracking systeM*. Through its graphical interface, this demonstrator also aimed to alert on the fact that privacy protection in the wireless era goes beyond the device address randomization. To support this statement, we described how *Venom* can be used to experiment privacy-preserving mechanisms, and prototyped one that does not require any actions from users on their devices to express their dissent to tracking.

In a second time, after passive attacks, we focused on the possibilities of an *active* attacker that exploits GATT profiles to link distinct random addresses generated by a single device. Relying on $dataset_{Active}$, we exhibited that the content of a GATT profile can be leveraged to fingerprint a device. Indeed, we found that identifiers and values composing this profile are diverse enough to act as a fingerprint.

As a conclusion of this first milestone, we proposed the first attempt at automatically

---

1. Used throughout this thesis, we draw the attention to the fact that those two datasets are not exclusively dedicated to study the physical tracking issue.

verifying the correctness of address randomization implementation through *Valkyrie* (*Verification of Addresses LinKabilitY in address Randomization ImplemEntations*), a versatile tool that is able to check properties written in a `Wireshark` based language. From previous works done by the community, we presented limitations of the address randomization as well as privacy properties of network traffic. Based on the definition of the frame unlinkability, we then designed a specific rules syntax, a verification process and an address reuse detection mechanism. To evaluate *Valkyrie*, we relied on a set of sixty different devices composed of laptops, watches and smartphones running across distinct OS versions. Finally, we asserted that this tool is able to detect issues in the generated wireless traffic.

Inferring users information from Bluetooth/BLE wireless communications: To follow our study on physical tracking, we examined how *passive* and *active* attackers can infer users information from the Bluetooth/BLE wireless communications. To this end, we established our threat model that was considered throughout this second milestone of this thesis, and identified *passive* attacks that could be set up. Moreover, it is important to mention that we limited our threat model to information that can be directly inferred from the observations of exposed data.

Leveraging $dataset_{Passive}$, we showed that the advertising data hold enough information to reveal the device model, type and manufacturer through the OUI, company IDs and service UUIDs, for instance. More worrisome, we found that such a device information can betray a medical condition of its corresponding user (e.g. a glucometer and an insulin pen both reveal diabetes). In the same vein, we observed that the identity of the owner under the form of its first name, last name or complete name can be included in the payload of advertisement packets raising serious privacy issues.

To complete our work, we analyzed *active* attacks over services, characteristics, and readable characteristics exposed within GATT profiles of $dataset_{Active}$. From our observations, we noticed that leaked information through identifiers and enumerated types can be leveraged for inventory attacks. Likewise, we made an unexpected discovery in which the LG manufacturer diverts Bluetooth SIG defined characteristics to include data that are not related to their initial purpose. As an example, the LG G5 and G6 smartphones use the `Altitude` Bluetooth SIG defined characteristic to carry a static device address. As such, this finding assesses the lack of control with regard to the release of connected objects.

Then, we depicted an activity inference attack that exploits a request-response mechanism of the L2CAP layer of the Bluetooth stack to infer the activity of a smartphone. Combining timing measurements with a change point detection analysis, we demonstrated that a nearby eavesdropper can be able to detect state changes of a device with a high accuracy.

Similarly to *Venom*, we wanted to raise public awareness about the information that can be inferred from the observations of the data contained within the advertisement packets and GATT profiles. As a consequence, we prototyped *Himiko*, a *Human Interface for Monitoring and Inferring Knowledge on Bluetooth Low Energy Objects*. Through this tool, we also highlighted that privacy considerations of the Bluetooth Core Specification have to be complemented.

Finally, from restricting the access to values of characteristics included within GATT profiles to enforcing a random back-off time in the L2CAP ping response process, we concluded this second milestone by furnishing recommendations that limit the impact of the outlined privacy threats.

The case of personal data leaks in *Apple* BLE Continuity protocols: In order to improve the illustration of the aforementioned physical tracking and inference of users information issues, we continued our research presenting a collection of personal data leaks that apply to the particular case of *Apple* BLE Continuity protocols. Given the popularity of the appliances produced by *Apple* and partner companies, we underline that hundreds of millions users are then currently impacted as the discovered concerns apply to all the 1.5 billion active *Apple* devices worldwide [199].

To open the third milestone of this thesis, we described the methodology that we used to thoroughly reverse engineer the suite of *Apple* Continuity protocols. Also, we discussed general features of their advertised messages such as their content protection and the use of random device addresses. Besides, we showed that 1) the payload of *Apple* Continuity messages is never encrypted as a whole and 2) some protocols such as AirPrint, AirDrop, "Hey Siri" and AirPlay use stable device addresses, thus exposing their users to tracking.

To pursue our work, we compiled a list of identifiers and counters included in broadcasted messages that can lead to *passive* tracking menaces. Beyond those artifacts, we found that battery levels and the `Lid Open Count` of AirPods earphones hold together a potential for tracking users.

Afterwards, we discovered a novel *active* tracking attack based on corrupted Handoff messages. Supplementing the Instant Hotspot replay attack introduced by Martin et al. [1], we emphasized that our attack is easier to exploit as Handoff messages are more common than Instant Hotspot ones.

To infer the activity and presence of an individual within its smarthome, we showed how a remote attacker can exploit increments of the `Global State Number` (`GSN`) embedded within HomeKit messages. Note that, we draw the attention to the fact that this attack does not allow the direct inference of a specific activity but exposes a coarse grain information (e.g. moving in a room, using a light, etc.).

Included within AirDrop and Nearby Action messages, we also demonstrated how hashed e-mail addresses and phone numbers can be recovered through *guesswork* attacks (i.e. brute force attacks for re-identification). As the prerequisite of those attacks, we selected different sets of identifiers from official and leaked databases, and described how to build a dictionary of practical size prior to simulate our attacks. From the experimental results, we asserted that recovering those hashed identifiers is possible without ambiguity in a matter of seconds when the set of potential value contains several thousand identifiers.

From our reverse engineering of *Apple* Continuity protocols, we uncovered that "Hey Siri"

messages embed a 2-byte long perceptual hash. To push our study further, we made some observations on the Siri's perceptual hashes at our disposal, and showcased that they depend on the command but on the user too. Indeed, the speaking speed, the voice tone as well as the pronunciation appear to have an influence on the generated digest. To evaluate our attack on such hashes, we built a dictionary of hundred commands and digests. For an exact match of the perceptual hashes, the results revealed that a precision as high as 67% with a recall of 52% can be obtained.

Last but not least, we appraised the impact of the reported privacy vulnerabilities. Relying on the feasibility of our presented attacks with regard to the range, the context in which they can be performed and the privacy risk resulting from the information that a potential attacker can collect, we attested that most of those attacks are straightforward to implement.

To finish, from the encryption and content minimization of continuity messages to the use of temporary random device addresses when advertising, we concluded this third milestone by providing recommendations that should be addressed by *Apple* to protect the privacy of their users.

As a side note, all the privacy issues identified during our research were responsibly disclosed to the following manufacturers: *AirBolt*, *Apple*, *Arcadyan*, *Bang&Olufsen*, *Boosted Boards*, *Bose*, *Diggro*, *Eve*, *Fitbit*, *Garmin*, *Giant*, *Google*, *GoPro*, *Jabra*, *Jewelbots*, *LG/LG Innotek*, *Logitech*, *Microsoft*, *Nikon*, *Nintendo*, *Nokia/Withings*, *OculusVR*, *Osram*, *Samsung*, *Sony*, *Tile* and *Xiaomi*.

VI.2   Limitations and perspectives

In this section, we detail the limitations of this thesis through a set of ten perspectives that could be applied to future work.

The pairing process of Bluetooth is based on key passing to identify devices and cipher communications. When a connection initiator or responder has `NoInputNoOutput` capabilities, the key generation method defaults to the unauthenticated `Just Works` mode [38, Vol 3, Part H, sec. 2.3.5.2]. As such, this means that devices such as smartphones and smartwatches with secure options have to downgrade their security to communicate with a `NoInputNoOutput` device. To extend our work, a first perspective could be then to leverage the possibilities offered by this downgrade attack to fetch IRKs (see Section I.2.2.2) of nearby devices in order to track their corresponding owners over time.

Briefly exploited, the *Microsoft* CDP and *Google* Nearby BLE proximity protocols merit as much attention as the *Apple* Continuity ones. In this regard, *Google* Nearby has been thoroughly reverse engineered by Antonioli et al. [234]. However, to the best of our knowledge, there is no similar work on the *Microsoft* CDP protocol. As a consequence,

an in-depth study on this protocol (and on proximity protocols in general [1]) could be a second perspective to consider.

De facto, the *Venom* demonstrator leverages the BD_ADDR and data included within advertisement packets to *passively* track devices. A third perspective could be to enrich this demonstrator with the GATT profiles fingerprinting technique (see Section III.4.2) to improve its tracking algorithm. Moreover, integrating privacy-preserving data structures as introduced by Alaggan et al. [236] could augment the system as well. Note that, we point out that *Venom* has been made to deploy and test privacy-enhancing features for tracking systems. To this purpose, we implemented a fully-functional platform that future research works could also reuse in order to experiment their custom privacy protection mechanisms.

Envisioned as a fourth perspective, we could develop a more generic automated version of the attempt at verifying the correctness of address randomization implementation that we provided in Section III.5. Indeed, we witnessed that the manual leakage search is cumbersome in practice, and engineers need a good understanding of protocols before to write new rules. In its current state, our proposed approach is thus mainly suitable for regression testing. Since all the vendor-specific extensions are hard to manually tackle, an automated approach would scale better. In fact, with enough network traffic captures from the same device types [2] for a lot of device types, a fully automated version could run as follows:

— if a field is the same (or related in a similar fashion, e.g. a counter) in all captures, then it is probably a fixed part of this packet type or structure;
— if a field is the same (or related) in all devices of a certain type, but different to other device types, then it is probably a feature that can be used to identify this device type; and in many cases uniquely link those devices, provided that there is only a limited number observed at a given time;
— if a field is the same (or related) to just one device, but different to other devices of the same type, then it is a unique feature and a clear linkability leakage.

Applying this process, our *Valkyrie* tool could then automatically work with never seen vendor extensions and consider the sequence of extension blocks too. Likewise, it would allow being easily ported to more packets and protocols, as long as there exist `Wireshark` dissectors.

As a fifth perspective, our Bluetooth based timing attack (see Section IV.4.3) could be evaluated on several different IoT devices. Beyond our theoretical discussion on its applicability to connected objects (see Section IV.4.3.6), this would allow to further appraise the impact of this attack on the privacy of users, while bringing results more statistically significance at the same time. In addition, we relied on detecting changes within solely seven device states. As a result, the exploration of more fine-grained inferences

---

1. With the multiplication of connected devices and their related applications/services, this perspective can be extended to all the existing proximity protocols.
2. But from different instances.

could benefit this work: for instance, by selecting other states that could leak web-surfing, use of GPS and downloading data.

During our study on *Apple* Continuity protocols, we showed that battery levels as well as the `Lid Open Count` of AirPods earphones can be used together to defeat the address randomization scheme. At the end of our demonstration, we highlighted that the global energy of the system could be exploited as well for tracking purposes (see Section V.3.2). As such, a sixth perspective could start with modeling the energy transfer between the case and the AirPods prior to create a global energy fingerprint that could be leveraged to track a set of AirPods between usage sessions.

In this same study, we detailed how perceptual hashes included within "Hey Siri" messages can lead a *passive* eavesdropper to infer spoken commands of a remote user. To this end, we observed that the values of such perceptual hashes depend on the speaker (see Section V.8.2). As a seventh perspective, we could tackle this issue by building a universal dictionary of commands and digests with the help of machine learning to mimic the voice of the target [228].

Overall, the contributions that we provided in this thesis are mainly focused on the BLE technology. A eighth perspective could be then to extend our research to other similar [1] IoT radio technologies such as Z-Wave and ZigBee. In particular, we can ask the question: "how our presented results on *physical tracking issues* and *inventory attacks* could be applied to other technologies ?". Additionally, evaluating the impact of our demonstrated attacks on wireless technologies such as LoRa and Sigfox could be considered too as their long range (i.e. several miles) could more expose devices to privacy breaches.

To defeat the device address randomization, a ninth perspective could be to correlate advertising data of BLE with data and metadata that other technologies such as Wi-Fi can furnish. As a reminder, this approach has been already introduced in Section IV.4.3.1 where we leveraged emitted Wi-Fi frames to obtain a BD_ADDR, even if the targeted Bluetooth device was not in discoverable mode.

Furthermore, to bring more material to such a ninth perspective, we could add the problem of connected vehicles which are likely to embed a plethora of wireless communicating sensors. More precisely, to provide insights on this future work, we made an experiment with Tire Pressure Monitoring Systems (TPMS). Note that, TPMS aim to monitor the air pressure inside the pneumatic tires on various types of vehicles. Deploying on each wheel of a vehicle, a TPMS sensor then sends its tire pressure data to TPMS receivers via Ultra High Frequency (UHF) radio [2]. Actually, additional information such as the TPMS serial number is also transmitted to the receivers. Acting as a unique and static identifier, it is important to mention that this information can be collected [3] with off-the-shelf hardware and open source software. As an illustration, Figure I.1 in Appendix I presents a draft of this

---

1. In terms of applications, bandwidth, power consumption and range.
2. Signals are transmitted at about 433 MHz in Europe (see Figure II.2), and at 315 MHz in most other parts of the world.
3. During our experiment, we witnessed a range of about fifty meters.

research where we can observe that collecting TPMS messages continuously broadcasted by our car[1] parked near our laboratory could lead to several privacy threatening applications such as *physical tracking* and *worker surveillance*. Moreover, by leveraging the static identifiers carried by such TPMS messages, it could be then possible to link the random device addresses generated by the BLE devices that belong to the driver and/or passengers of the car.

Lastly, a tenth and endless perspective would be to jointly work with regulation entities in order to 1) undermine the abuse of wireless technologies and 2) bring our technical point of views to public debate with regard to their privacy implications. In this direction, we described two main privacy vulnerabilities (i.e. *physical tracking* and *inference of users information*) throughout this thesis, and endeavored to give recommendations (see Section IV.6 and Section V.10) that manufacturers have to take into account during the design and implementation of their protocols to preserve the privacy of their users. In parallel, we draw the attention to the fact that this perspective can be linked to the implementation verification of the protocols and the providing of tools such as *Valkyrie* [21] to audit and/or certify such implementations.

VI.3 Impacts of our research

Outside the scientific community, we adopted a strategy to disseminate our results whose objective was to reach all stakeholders: from regulators to consumers, through industrials. Leveraging this approach, the impacts we had are detailed as follows.

Impact on the industry: The outcomes of this thesis have been discussed with industrials developing and/or deploying IoT products. More precisely, we were contacted by *Google* with regard to our paper [6] demonstrating that the BLE advertising data can be leveraged to defeat the LE Privacy feature. Actually, the general knowledge established around this issue gave rise to several meetings and interviews with the privacy engineers[2] of *Google* in order to understand how the identified pitfalls (i.e. static identifiers and non-reset counters in advertisement packets) might affect their software and devices.

As mentioned in Section VI.1, the privacy issues found during our research were responsibly[3] disclosed to twenty eight manufacturers including *Microsoft*, *Nokia/Withings*, *Samsung* and *Sony*. In fact, some of them such as *Microsoft* considered our findings and allowed us[4] to publish our results, others such as *Sony* redirected our requests to their own disclosure platform in order to register our discovered vulnerabilities but, thereafter, appeared to ignore our warnings, and most of the remaining manufacturers did not react at all[5].

---

1. To minimize the privacy implications that this experiment can raise, we only collected TPMS messages emitted by a car at our disposal.
2. Especially, with Eun-Jeong Shin [237].
3. We left manufacturers time to react to our solicitations.
4. After having reviewed their systems/protocols.
5. They did not even respond to our e-mails.

Note that, the responsible disclosure with regard to the case of the privacy leaks in *Apple* Continuity protocols is more particular. Indeed, we found major flaws and, more than a year later, they did nothing [1] to address such issues. Did *Apple* consider the reported privacy concerns as not serious ? Or are those concerns too complicated to fix ? Overall, this seems to be in contradiction with the *Apple* claimed motto [238]: *"privacy is a fundamental human right. At Apple, it's also one of our core values"*.

Finally, supporting the IoT chair at INSA Lyon, the SPIE ICS company has been privileged by having a direct access to our results. Indeed, regular meetings have assured that this industrial benefits from the developed expertise on privacy preservation within wireless communications. As a side note, we refer the interested readers to the white book [239] relating our exchanges in the context of this chair.

Popularization: Mathieu Cunche [50], one of the supervisors of this thesis, diffused the findings of our study at INSA Lyon through the master level courses *Sécurité et Vie Privée* [240] that is dedicated to the privacy protection.

In addition, to allow the reproduction of our research results, we put efforts to produce software [22, 21]. Besides, we point out that the Furious MAC research group at the United States Naval Academy relied [241] on our reverse engineering of *Apple* Continuity protocols (see Section V.2) to implement their custom `Wireshark` dissector [2]. Through a talk [242] that they give at *SchmooCon XVI* (an American hacker convention), they also participated to the popularization of our work on *Apple* Continuity.

Lastly, during the COVID-19 pandemic, we made our technical insights on the BLE available to French journalists [243, 244] to better inform the general public of the privacy implications that the use of such a technology can have, especially when it is employed in a proximity tracing mobile application.

Our work in the press: Our study on personal data leaks in *Apple* Continuity protocols [3] has been relayed a lot in the press. Indeed, several famous French Information and Communications Technology (ICT) websites [245, 246] as well as a popular science magazine [247] alerted the general public of our discoveries. Likewise, this work made the front page of *Le Progrès* [248], a daily newspaper that reports primarily on local news in the Rhône-Alpes region.

VI.4 Concluding remarks

In this ending section, we explain what we hope will happen to our research results.

---

1. At the exception of a minor improvement on the `Lid Open Count` of Proximity Pairing messages emitted by AirPods (see Section V.12).

2. Differing from the one they released, our `Wireshark` dissector [22] is as capable of parsing messages from *Apple* Continuity protocols as from additional BLE proprietary protocols such as the *Microsoft* CDP [148] and *Garmin* ones.

Beyond science, our objective is to raise public awareness through this thesis. Leveraging our findings along with the demonstrators and media coverage, we increased the visibility of the privacy issues associated with wireless communications of the IoT devices. Indeed, it is our belief that informed consumers will select more privacy friendly appliances forcing industrials to develop privacy preserving systems and, at the same time, inciting legislators as well as regulators to seriously consider those issues.

Having our results integrated into products and processes of industrials is also one of our expected outcome. To this end, we gave the industry access to software and experimental platforms in order to evaluate their products during the development phases. In particular, we draw the attention on the implemented approach within our *Valkyrie* tool (see Section III.5). As a part of a certification process, vendors could then rely on such a tool to verify that privacy properties are enforced by their devices. Note that, this joins the *privacy by design* approach advised by the GDPR and ePR regulations (see Section II.6.4).

Bluetooth/BLE and Wi-Fi are radio technologies that we focused in our work. In fact, those technologies are part of the IEEE 802 standards committee which specifies Wireless Personal Area Network (WPAN) standards. Currently, it is important to mention that such standardization bodies conduct efforts toward improving security and privacy protection when developing standards. In this regard, our work could benefit future radio standards in order to elaborate guidelines, ensure that appropriate measures are taken to correct critical flaws, and avoid privacy concerns such as those identified in this thesis.

From users privacy protection to general public information, government bodies such as the CNIL have an important role to play too. Respectively detailed in Section III.3.5 and Section IV.5, our *Venom* and *Himiko* demonstrators could be then suggested as an addition to the *Laboratoire d'Innovation Numérique de la CNIL* (LINC) [249].

To finish, Data Protection Authorities (DPA) could amplify the social impact of our research by reusing our findings. As an example, this could take the form of our results being cited within their opinions and regulatory documents such as the PIA of CNIL dedicated to the IoT [108], but also the opinion of the G29 (Working Party of Article 29) on the deployment of the IoT [109].

Bibliography

[1] Jeremy Martin, Douglas Alpuche, Kristina Bodeman, Lamont Brown, Ellis Fenske, Lucas Foppe, Travis Mayberry, Erik Rye, Brandon Sipes, and Sam Teplov. Handoff All Your Privacy – A Review of Apple's Bluetooth Low Energy Continuity Protocol. *Proceedings on Privacy Enhancing Technologies*, 2019(4):34–53, 2019.

[2] Bluetooth Special Interest Group. *GATT Specification Supplement v1.1*. 2019. URL https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=480006. Accessed: 2020-04-13.

[3] Guillaume Celosia and Mathieu Cunche. Discontinued Privacy: Personal Data Leaks in Apple Bluetooth-Low-Energy Continuity Protocols. *Proceedings on Privacy Enhancing Technologies*, 2020(1):26–46, 2020.

[4] Guillaume Celosia and Mathieu Cunche. Valkyrie: A Generic Framework for Verifying Privacy Provisions in Wireless Networks. In *Proceedings of the 13th Conference on Security and Privacy in Wireless and Mobile Networks*, pages 278–283, 2020.

[5] Guillaume Celosia and Mathieu Cunche. DEMO: Venom: a Visual and Experimental Bluetooth Low Energy Tracking System. In *Proceedings of the 13th Conference on Security and Privacy in Wireless and Mobile Networks*, pages 346–348, 2020.

[6] Guillaume Celosia and Mathieu Cunche. Saving Private Addresses: An Analysis of Privacy Issues in the Bluetooth-Low-Energy Advertising Mechanism. In *MobiQuitous 2019 – 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 444–453, 2019.

[7] Guillaume Celosia and Mathieu Cunche. DEMO: Himiko: A Human Interface for Monitoring and Inferring Knowledge on Bluetooth-Low-Energy Objects. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, pages 292–293, 2019.

[8] Guillaume Celosia and Mathieu Cunche. Detecting smartphone state changes through a Bluetooth based timing attack. In *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 154–159, 2018.

[9] Guillaume Celosia and Mathieu Cunche. Fingerprinting Bluetooth-Low-Energy Devices Based on the Generic Attribute Profile. In *Proceedings of the 2nd International*

ACM Workshop on Security and Privacy for the Internet-of-Things, pages 24–31, 2019.

[10] Guillaume Celosia and Mathieu Cunche. Saving Private Addresses: An Analysis of Privacy Issues in the Bluetooth-Low-Energy Advertising Mechanism. In *Atelier sur la Protection de la Vie Privée (APVP)*, 2019.

[11] Guillaume Celosia and Mathieu Cunche. Detecting smartphone state changes through a Bluetooth based timing attack. In *Atelier sur la Protection de la Vie Privée (APVP)*, 2018.

[12] Guillaume Celosia and Mathieu Cunche. Protection de la Vie Privée dans les Communications Sans Fil de l'Internet des Objets (IoT). In *Rendez-vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information (RESSI)*, 2018.

[13] Guillaume Celosia and Mathieu Cunche. POSTER: Preserving Privacy in Wireless Communications of the Internet of Things (IoT). In *Rendez-vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information (RESSI)*, 2018.

[14] Guillaume Celosia and Mathieu Cunche. Discontinued Privacy: Personal Data Leaks in Apple Bluetooth-Low-Energy Continuity Protocols. In *Annual seminar of the Inria PRIVATICS team*, 2020.

[15] Guillaume Celosia and Mathieu Cunche. DEMO: Himiko: A Human Interface for Monitoring and Inferring Knowledge on Bluetooth-Low-Energy Objects. In *Salon de l'Internet Des Objets (SIDO)*, 2019.

[16] Guillaume Celosia and Mathieu Cunche. Privacy Threats from the Bluetooth Low Energy Service Discovery Mechanism. In *Annual seminar of the Inria PRIVATICS team*, 2019.

[17] Guillaume Celosia and Mathieu Cunche. Detection of Bluetooth surveillance systems. In *Presentation for Inria EPI: Data and Algorithmic Transparency and Accountability (DATA)*, 2018.

[18] Guillaume Celosia and Mathieu Cunche. Privacy concerns in IoT radio communications. In *Pitch for the INSA Lyon – SPIE ICS IoT chair*, 2018.

[19] Mathieu Cunche, Célestin Matte, and Guillaume Celosia. Wi-Fi Scanner: How many data can be collected on your smartphone ? In *The Web Conference*, 2018.

[20] Guillaume Celosia and Mathieu Cunche. Preserving Privacy in Wireless Communications of the Internet of Things (IoT). In *Annual seminar of the Inria PRIVATICS team*, 2018.

[21] Guillaume Celosia and Mathieu Cunche. Valkyrie – A generic framework for verifying privacy provisions in wireless networks. 2020. URL https://github.com/celosiag/valkyrie. Accessed: 2020-04-02.

[22] Guillaume Celosia and Mathieu Cunche. Joker – A Wireshark dissector for Bluetooth Low Energy (BLE) advertisement packets of Apple Continuity, Microsoft CDP and Garmin proprietary protocols. 2020. URL https://github.com/celosiag/joker. Accessed: 2020-04-02.

[23] Statistica. Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025. 2016. URL https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide. Accessed: 2020-03-24.

[24] Naserddine Bouhaï, Imad Saleh, and Hakim Hachour. *Frontières numériques et savoir*. Editions L'Harmattan, 2016.

[25] Elisa Bertino. Data Security and Privacy in the IoT. In *EDBT*, volume 2016, pages 1–3, 2016.

[26] Jean-Marc Manach. *La vie privée, un problème de vieux cons?* FYP editions, 2010.

[27] Daniel J Solove. I've got nothing to hide and other misunderstandings of privacy. *San Diego L. Rev.*, 44:745, 2007.

[28] European Parliament. REPORT with recommendations to the Commission on Civil Law Rules on Robotics. 2017. URL https://www.europarl.europa.eu/doceo/document/A-8-2017-0005_EN.pdf. Accessed: 2020-05-11.

[29] UN General Assembly. Universal declaration of human rights. *UN General Assembly*, 302(2), 1948.

[30] Samuel D Warren and Louis D Brandeis. The right to privacy. *Harvard law review*, pages 193–220, 1890.

[31] Alan F Westin. Privacy and freedom. *Washington and Lee Law Review*, 25(1):166, 1968.

[32] Amira Barki, Abdelmadjid Bouabdallah, Said Gharout, and Jacques Traore. M2M security: Challenges and solutions. *IEEE Communications Surveys & Tutorials*, 18 (2):1241–1254, 2016.

[33] Hossein Shafagh, Anwar Hithnawi, Andreas Droescher, Simon Duquennoy, and Wen Hu. Talos: Encrypted query processing for the internet of things. In *Proceedings of the 13th ACM conference on embedded networked sensor systems*, pages 197–210, 2015.

[34] Sanaah Al Salami, Joonsang Baek, Khaled Salah, and Ernesto Damiani. Lightweight encryption for smart home. In *2016 11th International Conference on Availability, Reliability and Security (ARES)*, pages 382–388. IEEE, 2016.

[35] Nigel Davies, Nina Taft, Mahadev Satyanarayanan, Sarah Clinch, and Brandon Amos. Privacy mediators: Helping IoT cross the chasm. In *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications*, pages 39–44, 2016.

[36] Ricardo Neisse, Gary Steri, Igor Nai Fovino, and Gianmarco Baldini. SecKit: a model-based security toolkit for the internet of things. *computers & security*, 54: 60–76, 2015.

[37] Harry Chen, Tim Finin, Anupam Joshi, Lalana Kagal, Filip Perich, and Dipanjan Chakraborty. Intelligent agents meet the semantic web in smart spaces. *IEEE Internet computing*, 8(6):69–79, 2004.

[38] Bluetooth Special Interest Group. *Bluetooth Core Specification v5.2*. 2019. URL https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=478726. Accessed: 2020-03-31.

[39] Bluetooth Special Interest Group. *Bluetooth Core Specification v4.0*. 2010. URL https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=456433. Accessed: 2020-04-02.

[40] Martin Woolley. Bluetooth Technology Protecting Your Privacy. 2015. URL https://www.bluetooth.com/blog/bluetooth-technology-protecting-your-privacy. Accessed: 2020-05-12.

[41] IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture. *IEEE Std 802-2014 (Revision to IEEE Std 802-2001)*, pages 1–74, June 2014. doi: 10.1109/IEEESTD.2014.6847097.

[42] Bluetooth Special Interest Group. Generic Access Profile – Assigned Numbers. 2020. URL https://www.bluetooth.com/specifications/assigned-numbers/generic-access-profile. Accessed: 2020-05-12.

[43] Paul Leach, Michael Mealling, and Rich Salz. A universally unique identifier (uuid) urn namespace. 2005.

[44] Bluetooth Special Interest Group. GATT Specifications – GATT Services. 2020. URL https://www.bluetooth.com/specifications/gatt/services. Accessed: 2020-04-09.

[45] TransparenTech. Online UUID Generator. 2020. URL https://www.uuidgenerator.net. Accessed: 2020-05-12.

[46] Canonical. uuidgen – Create a new UUID value. 2019. URL http://manpages.ubuntu.com/manpages/focal/en/man1/uuidgen.1.html. Accessed: 2020-05-12.

[47] Bluetooth Special Interest Group. *GATT Specifications*. 2019. URL https://www.bluetooth.com/specifications/gatt. Accessed: 2020-04-01.

[48] Bluetooth Special Interest Group. Device Information – GATT Services. 2020. URL https://www.bluetooth.com/xml-viewer/?src=https://www.bluetooth.com/wp-content/uploads/Sitecore-Media-Library/Gatt/Xml/Services/org.bluetooth.service.device_information.xml. Accessed: 2020-05-13.

[49] Bluetooth Special Interest Group. Battery Service – GATT Services. 2020. URL https://www.bluetooth.com/xml-viewer/?src=https://www.bluetooth.com/wp-content/uploads/Sitecore-Media-Library/Gatt/Xml/Services/org.bluetooth.service.battery_service.xml. Accessed: 2020-05-13.

[50] Mathieu Cunche. Mathieu Cunche – Associate Professor, INSA-Lyon - CITI, Inria - Privatics. 2020. URL https://perso.citi-lab.fr/mcunche/index.html. Accessed: 2020-05-15.

[51] Guillaume Celosia. Guillaume Celosia – PhD Student @ CITI Lab - Inria. 2020. URL https://perso.citi-lab.fr/gcelosia. Accessed: 2020-05-15.

[52] John Kooker. Bluetooth, zigbee, and wibree: A comparison of wpan technologies. *CSE 237A*, 20, 2008.

[53] Eric Blossom. GNU radio: tools for exploring the radio frequency spectrum. *Linux journal*, 2004(122):4, 2004.

[54] Philippe Biondi and the Scapy community. Scapy – Packet crafting for Python2 and Python3. 2020. URL https://scapy.net. Accessed: 2020-05-25.

[55] Michael Ossmann. HackRF, an open source SDR platform. 2016. URL https://greatscottgadgets.com/hackrf. Accessed: 2020-05-25.

[56] Michael Ossmann. Ubertooth One: an open source Bluetooth test tool. 2016. URL https://greatscottgadgets.com/ubertoothone. Accessed: 2020-05-25.

[57] Brian Fung. How stores use your phone's WiFi to track your shopping habits. 2013. URL http://www.washingtonpost.com/blogs/the-switch/wp/2013/10/19/how-stores-use-your-phones-wifi-to-track-your-shopping-habits. Accessed: 2020-03-17.

[58] Vasilios Mavroudis and Michael Veale. Eavesdropping whilst you're shopping: Balancing personalisation and privacy in connected retail spaces. *arXiv preprint arXiv:1807.05381*, 2018.

[59] Kassem Fawaz, Kyu-Han Kim, and Kang G Shin. Protecting privacy of {BLE} device users. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 1205–1221, 2016.

[60] Sandra Siby, Rajib Ranjan Maiti, and Nils Ole Tippenhauer. Iotscanner: Detecting privacy threats in iot neighborhoods. In *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security*, pages 23–30, 2017.

[61] Noah Apthorpe, Dillon Reisman, and Nick Feamster. A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic. *arXiv preprint arXiv:1705.06805*, 2017.

[62] Marianne Bertrand and Emir Kamenica. Coming Apart? Cultural Distances in the United States Over Time. Technical report, National Bureau of Economic Research, 2018.

[63] Noah Apthorpe, Dillon Reisman, Srikanth Sundaresan, Arvind Narayanan, and Nick Feamster. Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic. *arXiv preprint arXiv:1708.05044*, 2017.

[64] Yapeng Wang, Xu Yang, Yutian Zhao, Yue Liu, and Laurie Cuthbert. Bluetooth positioning using RSSI and triangulation methods. In *2013 IEEE 10th Consumer Communications and Networking Conference (CCNC)*, pages 837–842. IEEE, 2013.

[65] Mozilla. Mozilla Location Service (MLS). 2020. URL https://location.services.mozilla.com. Accessed: 2020-05-25.

[66] Apple. Getting Started with iBeacon – Version 1.0. 2014. URL https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf. Accessed: 2020-05-25.

[67] ABM Musa and Jakob Eriksson. Tracking unmodified smartphones using wi-fi monitors. In *Proceedings of the 10th ACM conference on embedded network sensor systems*, pages 281–294, 2012.

[68] Le T Nguyen, Yu Seung Kim, Patrick Tague, and Joy Zhang. IdentityLink: user-device linking through visual and RF-signal cues. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 529–539, 2014.

[69] Edward Snowden and Bunnie Huang. Against the law: Countering lawful abuses of digital surveillance, 2016.

[70] Ishtiaq Rouf, Robert D Miller, Hossen A Mustafa, Travis Taylor, Sangho Oh, Wenyuan Xu, Marco Gruteser, Wade Trappe, and Ivan Seskar. Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study. In *USENIX Security Symposium*, volume 10, 2010.

[71] Dominic Milano. Content control: Digital watermarking and fingerprinting. *White Paper, Rhozet, a business unit of Harmonic Inc., http://www.rhozet.com/whitepapers/Fingerprinting-Watermarking.pdf, Last accessed May*, 30, 2012.

[72] Thomas Emmanual Varghese. Techniques for fraud monitoring and detection using application fingerprinting, May 27 2014. US Patent 8,739,278.

[73] David Formby, Preethi Srinivasan, Andrew Leonard, Jonathan Rogers, and Raheem A Beyah. Who's in Control of Your Control System? Device Fingerprinting for Cyber-Physical Systems. In *NDSS*, 2016.

[74] Vladimir Brik, Suman Banerjee, Marco Gruteser, and Sangho Oh. Wireless device identification with radiometric signatures. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 116–127, 2008.

[75] Adam C Polak, Sepideh Dolatshahi, and Dennis L Goeckel. Identifying wireless users via transmitter imperfections. *IEEE Journal on selected areas in communications*, 29(7):1469–1479, 2011.

[76] Tien Dang Vo-Huu, Triet Dang Vo-Huu, and Guevara Noubir. Fingerprinting Wi-Fi devices using software defined radios. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 3–14, 2016.

[77] Mathy Vanhoef, Célestin Matte, Mathieu Cunche, Leonardo S Cardoso, and Frank Piessens. Why MAC address randomization is not enough: An analysis of Wi-Fi network discovery mechanisms. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 413–424, 2016.

[78] Asaf Zomet and Shlomo Reuben Urbach. Privacy-aware personalized content for the smart home, October 22 2019. US Patent 10,453,098.

[79] Bluetooth Special Interest Group. Assigned Numbers for Baseband. 2020. URL https://www.bluetooth.com/specifications/assigned-numbers/baseband. Accessed: 2020-05-25.

[80] Dimitris Geneiatakis, Ioannis Kounelis, Ricardo Neisse, Igor Nai-Fovino, Gary Steri, and Gianmarco Baldini. Security and privacy issues for an IoT based smart home. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1292–1297. IEEE, 2017.

[81] Jeremy Martin, Travis Mayberry, Collin Donahue, Lucas Foppe, Lamont Brown, Chadwick Riggins, Erik C Rye, and Dane Brown. A study of MAC address randomization in mobile devices and when it fails. *Proceedings on Privacy Enhancing Technologies*, 2017(4):365–383, 2017.

[82] Ben Greenstein, Ramakrishna Gummadi, Jeffrey Pang, Mike Y Chen, Tadayoshi Kohno, Srinivasan Seshan, and David Wetherall. Can Ferris Bueller Still Have His Day Off? Protecting Privacy in the Wireless Era. In *HotOS*, 2007.

[83] T Scott Saponas, Jonathan Lester, Carl Hartung, Sameer Agarwal, Tadayoshi Kohno, et al. Devices That Tell on You: Privacy Trends in Consumer Ubiquitous Computing. In *USENIX Security Symposium*, pages 55–70, 2007.

[84] Marc Langheinrich. Privacy by design – Principles of privacy-aware ubiquitous systems. In *International conference on Ubiquitous Computing*, pages 273–291. Springer, 2001.

[85] Marco Gruteser and Dirk Grunwald. Enhancing location privacy in wireless LAN through disposable interface identifiers: a quantitative analysis. *Mobile Networks and Applications*, 10(3):315–325, 2005.

[86] Lee Hutchinson. iOS 8 to stymie trackers and marketers with MAC address randomization. *Ars Technica*, 2014.

[87] Android Open Source Project. Android 6.0 Changes. 2020. URL https://developer.android.com/about/versions/marshmallow/android-6.0-changes.html. Accessed: 2020-05-26.

[88] Katie Skinner and Jason Novak. Privacy and your app. In *apple worldwide dev. conf.(wwdc)*, 2015.

[89] Emmanuel Grumbach. iwlwifi: mvm: support random MAC address for scanning. *Linux commit effd05ac479b*, 2014.

[90] Winkey Wang. Wireless networking in Windows 10. In *Windows Hardware Engineering Community conference (WinHEC)*, 2015.

[91] 802.11aq-2018 – IEEE Standard for Information technology – Telecommunications and information exchange between systems Local and metropolitan area networks – Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Preassociation Discovery. Technical report, IEEE standard association, 2018. URL https://standards.ieee.org/standard/802_11aq-2018.html.

[92] Vijay Srinivasan, John Stankovic, and Kamin Whitehouse. Protecting your daily in-home activity information from a wireless snooping attack. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 202–211, 2008.

[93] Kenji Yoshigoe, Wei Dai, Melissa Abramson, and Alexander Jacobs. Overcoming invasion of privacy in smart home environment with synthetic packet injection. In *2015 TRON Symposium (TRONSHOW)*, pages 1–7. IEEE, 2015.

[94] Levent Demir, Mathieu Cunche, and Cédric Lauradoux. Analysing the privacy policies of Wi-Fi trackers. In *Proceedings of the 2014 workshop on physical analytics*, pages 39–44. ACM, 2014.

[95] Julien Freudiger. How talkative is your mobile device? An experimental study of Wi-Fi probe requests. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 1–6, 2015.

[96] Célestin Matte, Mathieu Cunche, Franck Rousseau, and Mathy Vanhoef. Defeating MAC address randomization through timing attacks. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 15–20, 2016.

[97] Johannes K Becker, David Li, and David Starobinski. Tracking anonymized bluetooth devices. *Proceedings on Privacy Enhancing Technologies*, 2019(3):50–65, 2019.

[98] Dave Evans. The internet of things: How the next evolution of the internet is changing everything. *CISCO white paper*, 1(2011):1–11, 2011.

[99] Margaret Rouse et al. Internet of things (IoT). *IoT Agenda*, 2016.

[100] Andrés Molina-Markham, Prashant Shenoy, Kevin Fu, Emmanuel Cecchet, and David Irwin. Private memoirs of a smart meter. In *Proceedings of the 2nd ACM workshop on embedded sensing systems for energy-efficiency in building*, pages 61–66, 2010.

[101] Jan Henrik Ziegeldorf, Oscar Garcia Morchon, and Klaus Wehrle. Privacy in the Internet of Things: threats and challenges. *Security and Communication Networks*, 7(12):2728–2742, 2014.

[102] Harald Sundmaeker, Patrick Guillemin, Peter Friess, and Sylvie Woelfflé. Vision and challenges for realising the Internet of Things. *Cluster of European Research Projects on the Internet of Things, European Commision*, 3(3):34–36, 2010.

[103] Janna Anderson and Lee Rainie. The Gurus Speak. *Pew Research Center*, 2014.

[104] Dominik Leibenger, Frederik Möllers, Anna Petrlic, Ronald Petrlic, and Christoph Sorge. Privacy challenges in the quantified self movement – an EU perspective. *Proceedings on privacy enhancing technologies*, 2016(4):315–334, 2016.

[105] Hossein Fereidooni, Tommaso Frassetto, Markus Miettinen, Ahmad-Reza Sadeghi, and Mauro Conti. Fitness trackers: fit for health but unfit for security and privacy. In *2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, pages 19–24. IEEE, 2017.

[106] Alastair R Beresford and Frank Stajano. Location privacy in pervasive computing. *IEEE Pervasive computing*, 2(1):46–55, 2003.

[107] European Union. Proposal for a Regulation on Privacy and Electronic Communications. 2017. URL https://ec.europa.eu/digital-single-market/en/news/proposal-regulation-privacy-and-electronic-communications. Accessed: 2020-03-24.

[108] Commission Nationale de l'Informatique et des Libertés. Privacy Impact Assessment (PIA) – Application to IoT devices. 2018. URL https://www.cnil.fr/sites/default/files/atoms/files/cnil-pia-piaf-connectedobjects-en.pdf. Accessed: 2020-03-24.

[109] Working Party 223. Opinion 8/2014 on the on Recent Developments on the Internet of Things. 2014. URL http://www.dataprotection.ro/servlet/ViewDocument?id=1088. Accessed: 2020-03-24.

[110] Qwant. Qwant – The search engine that respects your privacy. 2020. URL https://www.qwant.com. Accessed: 2020-05-27.

[111] Silver Eco. [Infographie] Groupe La Poste: les Français et les objets connectés. 2016. URL https://www.silvereco.fr/infographie-groupe-la-poste-les-francais-et-les-objets-connectes/3170704?wb48617274=D9DFD9B1. Accessed: 2020-03-24.

[112] Chris Baraniuk. Surveillance: The hidden ways you're tracked. 2014. URL https://www.bbc.com/future/article/20141027-the-hidden-ways-youre-tracked. Accessed: 2020-03-25.

[113] Barton Gellman and Ashkan Soltani. NSA tracking cellphone locations worldwide, Snowden documents show. *The Washington Post*, 4:2013, 2013.

[114] Jeremy Scahill and Glenn Greenwald. The NSA's secret role in the US assassination program. *The intercept*, 10:2014, 2014.

[115] Keaton Mowery and Hovav Shacham. Pixel perfect: Fingerprinting canvas in HTML5. *Proceedings of W2SP*, pages 1–12, 2012.

[116] Peter Eckersley. How unique is your web browser? In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 1–18. Springer, 2010.

[117] Célestin Matte. *Wi-Fi tracking: Fingerprinting attacks and counter-measures*. PhD thesis, 2017.

[118] Oscar D Lara and Miguel A Labrador. A survey on human activity recognition using wearable sensors. *IEEE communications surveys & tutorials*, 15(3):1192–1209, 2012.

[119] Xing Su, Hanghang Tong, and Ping Ji. Activity recognition with smartphone sensors. *Tsinghua science and technology*, 19(3):235–249, 2014.

[120] Narayanan C Krishnan and Diane J Cook. Activity recognition on streaming sensor data. *Pervasive and mobile computing*, 10:138–154, 2014.

[121] Ford-Long Wong and Frank Stajano. Location privacy in bluetooth. In *European Workshop on Security in Ad-hoc and Sensor Networks*, pages 176–188. Springer, 2005.

[122] Bram Bonné, Arno Barzan, Peter Quax, and Wim Lamotte. WiFiPi: Involuntary tracking of visitors at mass events. In *2013 IEEE 14th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pages 1–6. IEEE, 2013.

[123] Taher Issoufaly and Pierre Ugo Tournoux. BLEB: Bluetooth Low Energy Botnet for large scale individual tracking. In *2017 1st International Conference on Next Generation Computing Applications (NextComp)*, pages 115–120. IEEE, 2017.

[124] Dieter Oosterlinck, Dries F Benoit, Philippe Baecke, and Nico Van de Weghe. Bluetooth tracking of humans in an indoor environment: An application to shopping mall visits. *Applied geography*, 78:55–65, 2017.

[125] CB Insights. The Store Of The Future: 150+ Startups Transforming Brick-And-Mortar Retail In One Infographic. 2017. URL https://www.cbinsights.com/research/retail-store-tech-startups-2016. Accessed: 2020-04-06.

[126] Adam Meyers. Danger Close: Fancy Bear Tracking of Ukrainian Field Artillery Units. 2016. URL https://www.crowdstrike.com/blog/danger-close-fancy-bear-tracking-ukrainian-field-artillery-units. Accessed: 2020-04-06.

[127] Łukasz Olejnik. Privacy of London Tube Wifi Tracking. 2017. URL http://blog.lukaszolejnik.com/privacy-of-london-tube-wifi-tracking. Accessed: 2020-04-06.

[128] Constantine E Kontokosta and Nicholas Johnson. Urban phenology: Toward a real-time census of the city using Wi-Fi data. *Computers, Environment and Urban Systems*, 64:144–153, 2017.

[129] Android Open Source Project. Privacy: MAC Randomization. 2020. URL https://source.android.com/devices/tech/connect/wifi-mac-randomization. Accessed: 2020-04-06.

[130] Apple. About the security content of iOS 8. 2020. URL https://support.apple.com/en-us/HT201395. Accessed: 2020-04-06.

[131] Raspberry Pi Foundation. Teach, Learn, and Make with Raspberry Pi. 2020. URL https://www.raspberrypi.org. Accessed: 2020-05-08.

[132] Mini USB Wireless Bluetooth CSR V4.0 USB Dongle Adapter Dual Mode Device for Windows 10 8 7 Vista XP 32/64. 2020. URL https://www.amazon.com/Wireless-Bluetooth-Dongle-Adapter-Windows/dp/B01ITDZ2A2. Accessed: 2020-05-08.

[133] BlueZ. BlueZ – Official Linux Bluetooth protocol stack. 2020. URL http://www.bluez.org. Accessed: 2020-03-30.

[134] Ian Harvey. bluepy – A Bluetooth LE interface for Python. 2014. URL https://ianharvey.github.io/bluepy-doc/index.html. Accessed: 2020-03-30.

[135] Simone Margaritelli. This Is Not a Post About BLE, Introducing BLEAH. 2017. URL https://www.evilsocket.net/2017/09/23/This-is-not-a-post-about-BLE-introducing-BLEAH. Accessed: 2020-03-30.

[136] Scott Lester. The Emergence of Bluetooth Low Energy. 2015. URL https://www.contextis.com/blog/the-emergence-of-bluetooth-low-energy. Accessed: 2020-03-30.

[137] Mohamed Imran Jameel and Jeffrey Dungen. Low-power wireless advertising software library for distributed M2M and contextual IoT. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 597–602. IEEE, 2015.

[138] Institut National de la Statistique et des Etudes Economiques. Fichier des noms. 2018. URL https://www.insee.fr/fr/statistiques/3536630. Accessed: 2020-03-30.

[139] Institut National de la Statistique et des Etudes Economiques. Fichier des prénoms. 2019. URL https://www.insee.fr/fr/statistiques/2540004. Accessed: 2020-03-30.

[140] Apple. iBeacon. 2020. URL https://developer.apple.com/ibeacon. Accessed: 2020-03-30.

[141] Google. Eddystone. 2018. URL https://github.com/google/eddystone. Accessed: 2020-03-30.

[142] Radius Networks. AltBeacon Protocol Specification v1.0. 2015. URL https://github.com/AltBeacon/spec. Accessed: 2020-03-30.

[143] Aveek K Das, Parth H Pathak, Chen-Nee Chuah, and Prasant Mohapatra. Uncovering privacy leakage in ble network traffic of wearable fitness trackers. In *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications*, pages 99–104, 2016.

[144] Indra Mohan Chakravarti, Radha Govira Laha, and Jogabrata Roy. *Handbook of methods of applied statistics, volume 2*. Wiley, 1967.

[145] TW Kirkman. Statistics to use: Kolmogorov-Smirnov test. *College of Saint Benedict and Saint Johns University. Retrieved October*, 7:2008, 1996.

[146] Bluetooth Special Interest Group. *Bluetooth Core Specification Supplement v9*. 2019. URL https://www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc_id=480305. Accessed: 2020-03-31.

[147] Apple. Use Continuity to connect your Mac, iPhone, iPad, iPod touch, and Apple Watch. 2019. URL https://support.apple.com/en-ca/HT204681#handoff. Accessed: 2020-03-31.

[148] Microsoft. *[MS-CDP]: Connected Devices Platform Protocol Version 3*. 2019. URL https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-cdp/f5a15c56-ac3a-48f9-8c51-07b2eadbe9b4. Accessed: 2020-03-31.

[149] Google. Nearby. 2020. URL https://developers.google.com/nearby. Accessed: 2020-03-31.

[150] Google. Google Play services. 2020. URL https://play.google.com/store/apps/details?id=com.google.android.gms&hl=en. Accessed: 2020-05-08.

[151] Google. Nearby – Connections API. 2018. URL https://play.google.com/store/apps/details?id=com.google.android.gms&hl=en. Accessed: 2020-03-31.

[152] Google. Connectivity Samples Repository. 2020. URL https://github.com/android/connectivity-samples. Accessed: 2020-03-31.

[153] Google. Use Nearby to find Chromecast devices. 2020. URL https://support.google.com/chromecast/answer/7073953?hl=en-GB. Accessed: 2020-03-31.

[154] Célestin Matte and Mathieu Cunche. Wombat: An experimental wi-fi tracking system. 2017.

[155] Sergey Bratus, Cory Cornelius, David Kotz, and Daniel Peebles. Active behavioral fingerprinting of wireless devices. In *Proceedings of the first ACM conference on Wireless network security*, pages 56–61, 2008.

[156] Gábor György Gulyás, Gergely Acs, and Claude Castelluccia. Near-optimal fingerprinting with constraints. *Proceedings on Privacy Enhancing Technologies*, 2016(4): 470–487, 2016.

[157] Apple. Secure software updates overview. 2020. URL https://support.apple.com/fr-fr/guide/security/secf683e0b36/web. Accessed: 2020-06-05.

[158] Marc Haase, Matthias Handy, et al. BlueTrack – Imperceptible tracking of bluetooth devices. In *Ubicomp Poster Proceedings*, page 2, 2004.

[159] Thomas Liebig and Armel Ulrich Kemloh Wagoum. Modelling Microscopic Pedestrian Mobility using Bluetooth. In *ICAART (2)*, pages 270–275, 2012.

[160] Mathias Versichele, Tijs Neutens, Matthias Delafontaine, and Nico Van de Weghe. The use of Bluetooth for analysing spatiotemporal dynamics of human movement at mass events: A case study of the Ghent Festivities. *Applied Geography*, 32(2): 208–220, 2012.

[161] Naeim Abedi, Ashish Bhaskar, and Edward Chung. Bluetooth and Wi-Fi MAC address based crowd data collection and monitoring: benefits, challenges and enhancement. 2013.

[162] Andreas Pfitzmann and Marit Hansen. Anonymity, unlinkability, unobservability, pseudonymity, and identity management-a consolidated proposal for terminology. 2005.

[163] KimiNewt. PyShark – Python packet parser using wireshark's tshark. 2020. URL https://kiminewt.github.io/pyshark. Accessed: 2020-04-02.

[164] Gerald Combs. Wireshark – The world's foremost and widely-used network protocol analyzer. 1998. URL https://www.wireshark.org. Accessed: 2020-04-02.

[165] Furious MAC Wireless Security & Privacy Research Group. Furious MAC is a project to understand, map, and correlate wireless hardware identifiers. 2020. URL https://furiousmac.com. Accessed: 2020-06-04.

[166] Giles Hogben. Changes to Device Identifiers in Android O. 2017. URL https://android-developers.googleblog.com/2017/04/changes-to-device-identifiers-in.html. Accessed: 2020-04-02.

[167] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *2013 IEEE Symposium on Security and Privacy*, pages 541–555. IEEE, 2013.

[168] Bluetooth Special Interest Group. 2020 Bluetooth Market Update. Technical report, 2020. URL https://www.bluetooth.com/wp-content/uploads/2020/03/2020_Market_Update-EN.pdf. Accessed: 2020-04-17.

[169] John Hering. The BlueSniper 'rifle'. *12th DEFCON, Las Vegas*, 2004.

[170] Adam Laurie, Marcel Holtmann, and Martin Herfurt. Bluetooone. 2004. URL https://trifinite.org/trifinite_stuff_bluetooone.html. Accessed: 2020-04-08.

[171] Bluetooth Special Interest Group. Company Identifiers – Assigned Numbers. 2020. URL https://www.bluetooth.com/specifications/assigned-numbers/company-identifiers. Accessed: 2020-04-09.

[172] Jeremy Martin, Erik Rye, and Robert Beverly. Decomposition of MAC address structure for granular device inference. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 78–88, 2016.

[173] Google. Google Shopping: Buy on Google | Get good deals from your merchants. 2020. URL https://shopping.google.com. Accessed: 2020-06-15.

[174] Amazon. Amazon.com: Online Shopping for Electronics, Apparel, Computers, Books, DVDs & more. 2020. URL https://www.amazon.com. Accessed: 2020-06-15.

[175] Bluetooth Special Interest Group. 16 Bit UUIDs for Members – Assigned Numbers. 2020. URL https://www.bluetooth.com/specifications/assigned-numbers/16-bit-uuids-for-members. Accessed: 2020-04-09.

[176] Android Open Source Project. Hearing Aid Audio Support Using Bluetooth LE. 2020. URL https://source.android.com/devices/bluetooth/asha. Accessed: 2020-04-09.

[177] Felipe Barriga. Open API of Sony SmartBand SWR-10. 2015. URL https://github.com/fbarriga/sony-smartband-open-api. Accessed: 2020-04-09.

[178] Facebook. Facebook – Log in or sign up. 2020. URL https://www.facebook.com. Accessed: 2020-06-15.

[179] Instagram. Instagram. 2020. URL https://www.instagram.com. Accessed: 2020-06-15.

[180] Twitter. Explore / Twitter. 2020. URL https://twitter.com. Accessed: 2020-06-15.

[181] Łukasz Olejnik, Gunes Acar, Claude Castelluccia, and Claudia Diaz. The leaking battery. In *Data Privacy Management, and Security Assurance*, pages 254–263. Springer, 2015.

[182] Jay Freeman. Models – The iPhone Wiki. 2020. URL https://www.theiphonewiki.com/wiki/Models. Accessed: 2020-04-13.

[183] Apple. Identify your MacBook Pro model. 2019. URL `https://support.apple.com/en-us/HT201300`. Accessed: 2020-04-13.

[184] USB Implementers Forum. Membership Lookup & List. 2020. URL `https://www.usb.org/members`. Accessed: 2020-04-13.

[185] Apple. Apple Notification Center Service (ANCS) Specification. 2014. URL `https://developer.apple.com/library/archive/documentation/CoreBluetooth/Reference/AppleNotificationCenterServiceSpecification/Specification/Specification.html#//apple_ref/doc/uid/TP40013460-CH1-SW8`. Accessed: 2020-04-13.

[186] Andrey Nikishaev. MiBand2 – Library to work with Xiaomi MiBand 2. 2020. URL `https://github.com/creotiv/MiBand2`. Accessed: 2020-04-13.

[187] Michael Ossmann. Discovering the Bluetooth UAP. 2014. URL `http://ubertooth.blogspot.fr/2014/06/discovering-bluetooth-uap.html`. Accessed: 2020-04-13.

[188] Michèle Basseville, Igor V Nikiforov, et al. *Detection of abrupt changes: theory and application*, volume 104. prentice Hall Englewood Cliffs, 1993.

[189] Wes McKinney. pandas – Python Data Analysis Library. 2020. URL `https://pandas.pydata.org`. Accessed: 2020-04-13.

[190] Paulo Borges. l2ping.c – BlueZ – Bluetooth protocol stack for Linux. 2012. URL `https://github.com/pauloborges/bluez/blob/master/tools/l2ping.c`. Accessed: 2020-06-15.

[191] Apple. Shazam. 2020. URL `https://www.shazam.com/apps`. Accessed: 2020-05-08.

[192] Apple. BLEScanner 4.0. 2020. URL `https://apps.apple.com/us/app/ble-scanner-4-0/id1221763603`. Accessed: 2020-05-08.

[193] Al Fidel. E-cigarette is better with Bluetooth. 2019. URL `https://vapebiz.net/e-cigarette-is-better-with-bluetooth`. Accessed: 2020-06-12.

[194] Bluetooth Special Interest Group. Specifications – Assigned Numbers. 2020. URL `https://www.bluetooth.com/specifications/assigned-numbers`. Accessed: 2020-04-10.

[195] Mozilla. Download Firefox – Free Web Browser. 2020. URL `https://www.mozilla.org/en-US/firefox/new`. Accessed: 2020-06-16.

[196] Apple. All your devices. One seamless experience. 2020. URL `https://www.apple.com/macos/continuity`. Accessed: 2020-04-21.

[197] Apple. MFi Program. 2020. URL `https://developer.apple.com/programs/mfi`. Accessed: 2020-04-21.

[198] Apple. Home accessories. The list keeps getting smarter. 2020. URL https://www.apple.com/ios/home/accessories. Accessed: 2020-04-21.

[199] Apple. Apple Reports Record First Quarter Results. 2020. URL https://www.apple.com/newsroom/2020/01/apple-reports-record-first-quarter-results. Accessed: 2020-04-21.

[200] Mathieu Cunche, Mohamed-Ali Kaafar, and Roksana Boreli. Linking wireless devices using information contained in Wi-Fi probe requests. *Pervasive and Mobile Computing*, 11:56–69, 2014.

[201] Apple. Apple Platform Security. 2020. URL https://support.apple.com/guide/security/welcome/web. Accessed: 2020-04-21.

[202] Apple. HomeKit Accessory Protocol Specification (Non-Commercial Version) – Release R2. 2019. URL https://developer.apple.com/homekit/specification. Accessed: 2020-04-21.

[203] Paulo Borges. hcitool.c – BlueZ – Bluetooth protocol stack for Linux. 2012. URL https://github.com/pauloborges/bluez/blob/master/tools/hcitool.c. Accessed: 2020-04-21.

[204] Apple. Apple Developer. 2020. URL https://developer.apple.com. Accessed: 2020-04-21.

[205] Vincent Bénony. Hopper v4 – The macOS and Linux Disassembler. 2020. URL https://www.hopperapp.com. Accessed: 2020-04-21.

[206] Apple. AirDrop security. 2020. URL https://support.apple.com/guide/security/sec2261183f4/web. Accessed: 2020-04-21.

[207] Apple. Handoff. 2020. URL https://support.apple.com/guide/security/secf78dbe639/web. Accessed: 2020-04-21.

[208] Apple. About AirPrint. 2020. URL https://support.apple.com/en-us/HT201311. Accessed: 2020-04-21.

[209] Ang Cui, Michael Costello, and Salvatore Stolfo. When firmware modifications attack: A case study of embedded exploitation. 2013.

[210] Apple. Connect and use your AirPods and AirPods Pro. 2020. URL https://support.apple.com/en-us/HT207010. Accessed: 2020-04-21.

[211] Dorene Kewley, Russ Fink, John Lowry, and Mike Dean. Dynamic approaches to thwart adversary intelligence gathering. In *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*, volume 1, pages 176–185. IEEE, 2001.

[212] Apple. Handoff. 2020. URL https://developer.apple.com/handoff. Accessed: 2020-04-21.

[213] Apple. Use Handoff to continue a task on your other devices. 2019. URL `https://support.apple.com/en-us/HT209455`. Accessed: 2020-04-21.

[214] Apple. Instant Hotspot. 2020. URL `https://support.apple.com/guide/security/seca4b33e8c9/web`. Accessed: 2020-04-21.

[215] Apple. Use Instant Hotspot to connect to your Personal Hotspot without entering a password. 2019. URL `https://support.apple.com/en-us/HT209459`. Accessed: 2020-04-21.

[216] Matthias C Sala, Kurt Partridge, Linda Jacobson, et al. An exploration into activity-informed physical advertising using PEST. In *International Conference on Pervasive Computing*, pages 73–90. Springer, 2007.

[217] Bogdan Copos, Karl Levitt, Matt Bishop, and Jeff Rowe. Is anybody home? Inferring activity from smart home network traffic. In *2016 IEEE Security and Privacy Workshops (SPW)*, pages 245–251. IEEE, 2016.

[218] Joseph Bonneau. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *2012 IEEE Symposium on Security and Privacy*, pages 538–552. IEEE, 2012.

[219] Levent Demir, Amrit Kumar, Mathieu Cunche, and Cedric Lauradoux. The pitfalls of hashing for privacy. *IEEE Communications Surveys & Tutorials*, 20(1):551–565, 2017.

[220] Matthias Marx, Ephraim Zimmer, Tobias Mueller, Maximilian Blochberger, and Hannes Federrath. Hashing of personally identifiable information is not sufficient. *SICHERHEIT 2018*, 2018.

[221] Troy Hunt. The 773 Million Record "Collection #1" Data Breach. 2019. URL `https://www.troyhunt.com/the-773-million-record-collection-1-data-reach`. Accessed: 2020-04-21.

[222] Jennifer Elias and Magdalena Petrova. Google's rocky path to email domination. 2019. URL `https://www.cnbc.com/2019/10/26/gmail-dominates-consumer-email-with-1point5-billion-users.html`. Accessed: 2020-04-21.

[223] PETS. Symposium Attendance – PoPETs Acceptance Rates. 2020. URL `https://petsymposium.org/acceptance-rates.php#attendance`. Accessed: 2020-04-21.

[224] Jens Steube and Gabriele Gristina. hashcat – Advanced password recovery. 2020. URL `https://hashcat.net/hashcat`. Accessed: 2020-04-21.

[225] Autorité de Régulation des Communications Electroniques et des Postes. Marché des communications électroniques en France (T4 2019). 2020. URL `https://www.arcep.fr/fileadmin/cru-1582218129/reprise/observatoire/4-2019/obs-marches-services-T4_2019-020420.pdf`. Accessed: 2020-04-21.

[226] Jaap Haitsma and Ton Kalker. A Highly Robust Audio Fingerprinting System. In *Ismir*, volume 2002, pages 107–115, 2002.

[227] Heiko Knospe. Privacy-enhanced perceptual hashing of audio data. In *2013 International Conference on Security and Cryptography (SECRYPT)*, pages 1–6. IEEE, 2013.

[228] Gopala Krishna Anumanchipalli, Kishore Prahallad, and Alan W Black. Festvox: Tools for creation and analyses of large speech corpora. In *Workshop on Very Large Scale Phonetics Research, UPenn, Philadelphia*, page 70, 2011.

[229] Sparhandy. Siri commands – Endless functions of your virtual assistant. 2020. URL https://www.sparhandy.de/apple/info/siri-commands. Accessed: 2020-04-21.

[230] Hao Fu, Aston Zhang, and Xing Xie. Effective social graph deanonymization based on graph structure and descriptive information. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(4):1–29, 2015.

[231] Jon Gunnar Sponas. Things You Should Know About Bluetooth Range. 2018. URL https://blog.nordicsemi.com/getconnected/things-you-should-know-about-bluetooth-range. Accessed: 2020-04-21.

[232] Nina Gerber, Benjamin Reinheimer, and Melanie Volkamer. Investigating People's Privacy Risk Perception. *Proceedings on Privacy Enhancing Technologies*, 2019(3): 267–288, 2019.

[233] Chrisil Arackaparambil, Sergey Bratus, Anna Shubina, and David Kotz. On the reliability of wireless fingerprinting using clock skews. In *Proceedings of the third ACM conference on Wireless network security*, pages 169–174, 2010.

[234] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. Nearby Threats: Reversing, Analyzing, and Attacking Google's 'Nearby Connections' on Android. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, February 2019.

[235] Milan Stute, Sashank Narain, Alex Mariotto, Alexander Heinrich, David Kreitschmann, Guevara Noubir, and Matthias Hollick. A billion open interfaces for Eve and Mallory: MitM, DoS, and tracking attacks on iOS and macOS through Apple Wireless Direct Link. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 37–54, 2019.

[236] Mohammad Alaggan, Mathieu Cunche, and Marine Minier. Privacy-Preserving t-Incidence for WiFi-based Mobility Analytics. 2016.

[237] Eun-Jeong Shin. Eun-Jeong Shin – Privacy Engineer of Google. 2020. URL https://scholar.google.com/citations?hl=en&user=nLwatNUAAAAJ. Accessed: 2020-06-24.

[238] Apple. Privacy – Apple. 2020. URL https://www.apple.com/privacy. Accessed: 2020-06-18.

[239] SPIE ICS and INSA Lyon. IoT & Privacy – Comment assurer la confidentialité sur les réseaux sans fil ? L'exemple du BLE. 2020. URL https://chaires.insa-lyon.fr/sites/chaires.insa-lyon.fr/files/lb4-iot_security_20200121_bat.pdf. Accessed: 2020-05-06.

[240] Mathieu Cunche. Télécommunications – Sécurité et vie privée. 2020. URL http://planete.insa-lyon.fr/scolpeda/f/ects?id=33882&_lang=fr. Accessed: 2020-05-06.

[241] Furious MAC research group. An Apple Continuity Protocol Reverse Engineering Project. 2020. URL https://github.com/furiousMAC/continuity. Accessed: 2020-06-19.

[242] Jeremy Martin, Douglas Alpuche, Kristina Bodeman, Lamont Brown, Ellis Fenske, Lucas Foppe, Travis Mayberry, Erik C. Rye, Brandon Sipes, and Sam Teplov. Reverse Engineering Apple's BLE Continuity Protocol for Tracking, OS Fingerprinting, and Behavioral Profiling. In *ShmooCon XVI*, 2020.

[243] Sophian Fanen. StopCovid : questions sur un mouchard de poche. 2020. URL https://lesjours.fr/obsessions/coronavirus-quarantaine/ep51-appli-deconfinement. Accessed: 2020-05-06.

[244] François Tonic. Programmez! podcast 17 : tracking et Covid19. 2020. URL https://podcast.ausha.co/poddev/programmez-podcast-16-tracking-et-covid19. Accessed: 2020-05-06.

[245] Florian Innocente. Bluetooth : Continuité n'est pas assez étanche avec l'échange de données. 2019. URL https://www.macg.co/macos/2019/12/bluetooth-continuite-nest-pas-assez-etanche-avec-lechange-de-donnees-110473. Accessed: 2020-05-06.

[246] Medhi Naitmazi. Continuity d'Apple : une fuite du Bluetooth permet de récupérer vos informations. 2019. URL https://iphonesoft.fr/2019/12/17/continuity-apple-fuite-bluetooth-permet-recuperer-informations. Accessed: 2020-05-06.

[247] Olivier Hertel. Des chercheurs français découvrent des fuites d'informations sensibles sur les appareils Apple. 2020. URL https://www.sciencesetavenir.fr/high-tech/conso/des-chercheurs-francais-decouvrent-des-fuites-d-informations-sensibles-sur-les-appareils-apple_139877. Accessed: 2020-05-06.

[248] Damien Lepetitgaland. Connexions Apple : un étudiant de l'Insa Lyon détecte des failles. 2020. URL https://www.leprogres.fr/edition-lyon-villeurbanne/2019/12/19/connexions-apple-l-etudiant-de-l-insa-detecte-des-failles. Accessed: 2020-05-06.

[249] Commission Nationale de l'Informatique et des Libertés. LINC – Laboratoire d'Innovation Numérique de la CNIL. 2020. URL https://linc.cnil.fr. Accessed: 2020-05-08.

Appendices

Appendix A

Example output of *Venom*
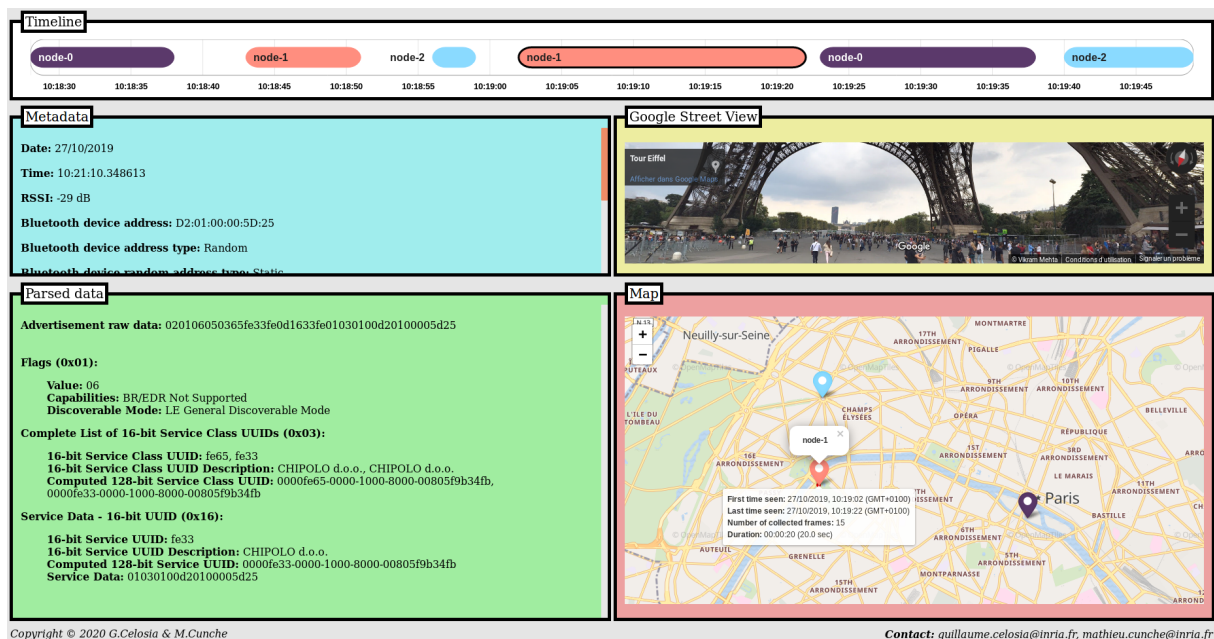
Figure A.1 – Example output of a *Chipolo* Classic BLE keyring device.

Appendix B

Example of a GATT profile from an *Apple* iPhone 8 smartphone

```json
{
    "metadata" : {
        "timestamp" : "40365"
    }
    "header" : {
        "pdu_type" : "ADV_IND",
        "bd_addr_type" : "random"
    }
    "payload" : {
        "bd_addr" : "4a:af:85:51:42:b4",
        "service_0" : {
            "handle_start" : "0001",
            "handle_end" : "0005",
            "name" : "Generic Access",
            "UUID" : "00001800-0000-1000-8000-00805f9b34fb",
            "characteristics" : [
                {
                    "handle" : "0003",
                    "name" : "Device Name",
                    "UUID" : "00002a00-0000-1000-8000-00805f9b34fb",
                    "properties" : "READ",
                    "value" : "iPhone"
                },
                {
                    "handle" : "0005",
                    "name" : "Appearance",
                    "UUID" : "00002a01-0000-1000-8000-00805f9b34fb",
                    "properties" : "READ",
                    "value" : "Generic Phone"
                }
            ]
        },
        "service_1" : {
            "handle_start" : "0006",
            "handle_end" : "0009",
            "name" : "Generic Attribute",
            "UUID" : "00001801-0000-1000-8000-00805f9b34fb",
            "characteristics" : [
                {
```

```
                    "handle" : "0008",
                    "name" : "Service Changed",
                    "UUID" : "00002a05-0000-1000-8000-00805f9b34fb",
                    "properties" : "INDICATE",
                    "value" : ""
                }
            ]
        },
        "service_2" : {
            "handle_start" : "000a",
            "handle_end" : "000e",
            "name" : "Apple Continuity Service",
            "UUID" : "d0611e78-bbb4-4591-a5f8-487910ae4366",
            "characteristics" : [
                {
                    "handle" : "000c",
                    "name" : "Continuity Characteristic",
                    "UUID" : "8667556c-9a37-4c91-84ed-54ee27d90049",
                    "properties" : "NOTIFY WRITE EXTENDED PROPERTIES",
                    "value" : ""
                }
            ]
        },
        "service_3" : {...}, <-- Apple Nearby Service
        "service_4" : {...}, <-- Battery Service
        "service_5" : {...}, <-- Current Time Service
        "service_6" : {
            "handle_start" : "001e",
            "handle_end" : "0022",
            "name" : "Device Information",
            "UUID" : "0000180a-0000-1000-8000-00805f9b34fb",
            "characteristics" : [
                {
                    "handle" : "0020",
                    "name" : "Manufacturer Name String",
                    "UUID" : "00002a29-0000-1000-8000-00805f9b34fb",
                    "properties" : "READ",
                    "value" : "Apple Inc."
                },
                {
                    "handle" : "0022",
                    "name" : "Model Number String",
                    "UUID" : "00002a24-0000-1000-8000-00805f9b34fb",
                    "properties" : "READ",
                    "value" : "iPhone10,4"
                }
            ]
        },
        "service_7" : {...}, <-- Apple Notification Center Service
        "service_8" : {...}  <-- Apple Media Service
    }
}
```

Figure B.1 – Example of a GATT profile collected from an *Apple* iPhone 8 smartphone (and formatted as a JSON string).

Appendix C

Device names, 16-bit and 128-bit service UUIDs found in BLE advertisement packets

Table C.1 – Extended list of experimental regular expressions of device names.

Device name	Description		
Regular expression	Manufacturer	Type of device	Model
^Alta HR$	Fitbit	Fitness tracker	Alta HR
^Ambit.	Suunto	Fitness tracker	Ambit
^Approach	Garmin	Fitness tracker	Approach
^AuraBox	Divoom	Speaker	AuraBox
^Beo[P,p]lay	Bang&Olufsen	Speaker	Beoplay
^BLE_Edge 1000	Garmin	Cycling GPS	Edge 1000
^DuoTrap	Bontrager	Cycling sensor	DuoTrap
^Edge	Garmin	Cycling GPS	Edge
^FDR-X3000$	Sony	Camera	FDR-X3000
^GALAXY Gear	Samsung	Smartwatch	Galaxy Gear
^honor Band	Huawei	Fitness tracker	Honor Band
^Ionic$	Fitbit	Fitness tracker	Ionic
^Jabra PULSE Smart	Jabra	Earphones	Pulse
^KTULU	Divacore	Speaker	Ktulu
^LE_GTK-XB..	Sony	Speaker	GTK-XB
^LeGaluchon$	Galanck	Backpack	Le Galuchon
^MIScooter	Xiaomi	Electric scooter	Mi Scooter
^PHANTOM$	Devialet	Speaker	Phantom
^Ride Sense	Giant	Cycling sensor	RideSense
^Rider.	Bryton	Cycling GPS	Rider
^Shine	Misfit	Fitness tracker	Shine
^The Dash	Bragi	Earphones	The Dash
^VivoWatch	Asus	Smartwatch	VivoWatch
^W Activite	Nokia/Withings	Smartwatch	Activite
^ZeRound	MyKronoz	Smartwatch	ZeRound

Table C.2 – Extended list of experimental 16-bit service UUIDs.

16-bit UUID	Description		
	Manufacturer	Type of device	Model
0x0f3e	TrackR	Keyring	Bravo
0x1600	Lapa	Keyring	Tracker
0x1725	Scosche	Fitness tracker	Rhythm+
0x2014	Cardo	Motorbike headset	Scala Rider
0x2220	Sylvania	Car light	ZEVO
0x261f	Livescribe	Pen	Smartpen
0x3031	Inexive	Fitness tracker	SmartBand
0x9900	Samsung	Earphones	Gear IconX
0xa000	HTC	Camera	RE Camera
0xacc0	Bionic bird	Biomimetic drone	Bird
0xff02	Mipow	Lightbulb	Playbulb Candle
0xff20	Wonlex	Fitness tracker	I5

Table C.3 – Extended list of experimental 128-bit service UUIDs.

128-bit UUID	Description		
	Manufacturer	Type of device	Model
00000200-37cb-11e3-8682-0002a5d5c51b	Sony	Fitness tracker	SmartBand SWR-10
00001859-73bf-11e3-9359-00186b00bcbc	LG	Fitness tracker	LifeBand Touch
00001c00-d102-11e1-9b23-00025b00c6c6	Nonda	Car charger	ZUS
00010000-f619-54a4-95e5-072a926cc46f	Drust	Car sensor	Akolyt
06aa1910-f22a-11e3-9daa-0002a5d5c51b	Nespresso	Coffee machine	Prodigio
0724ff01-0159-483b-ba9f-608b716c253e	Fitbug	Fitness tracker	Orb
14eb3500-785f-0000-0000-0401f4ac4ea4	XY Findit	Keyring	XY4+
151c1000-4580-4111-9ca1-5056f3454fbc	Jawbone	Fitness tracker	UP
3c63cc60-364a-11e3-808b-0002a5d5c51b	Moulinex	Cooker	Cookeo
3e400001-b5a3-f393-e0a9-e50e24dcca9e	Snapchat	Glasses	Spectacles
4553867f-f809-49f4-aefc-e190a1f459f3	Philips	Sleep monitor	Respironics
45855422-6565-4cd7-a2a9-fe8af41b85e8	Hidrate	Water bottle	Spark
61353090-8231-49cc-b57a-886370740041	Suunto	Fitness tracker	MoveSense
70d4f432-e9a9-412e-bc09-e8dc113dcdb9	Basis	Smartwatch	Peak
72daa6c3-29c2-6283-0c4a-2818e4d37e75	Logitech	Earphones	Ultimate Ears
83cdc410-31dd-11e2-81c1-0800200c9a66	Nike	Fitness tracker	Fuelband
84c80001-4a61-60b9-3a2b-1300855e588c	Giant	Cycling sensor	RideSense
a5f5c1e4-968f-11e6-ae22-56b6b6499611	Krups	Coffee machine	Evidence
aa745be2-9025-4bf2-a318-91f3dba2999f	Garmin	GPS	Nuvi
adab79d0-6e7d-4601-bda2-bffaa68956ba	Polaroid	Printer	Zip
b993bf90-81e1-11e4-b4a9-0800200c9a66	TomTom	Smartwatch	GPS Watch
d7634335-46a3-0c27-868b-598e8e218bfe	Parrot	Handsfree car kit	Minikit
d839fc3c-84dd-4c36-9126-187b07255126	SumUp	Payment terminal	Air Plus
dcd68980-aadc-11e1-a22a-0002a5d5c51b	Adonit	Pen	Stylus
f27a4900-92d6-49bf-8ac0-a661ccfbcb37	Invoxia	GPS tracker	Roadie

Appendix D

AD types distribution in $dataset_{Passive}$

In average, we found that advertising payloads within $dataset_{Passive}$ embed two distinct AD structures. Moreover, the distribution of those AD types is concentrated as sixteen types (see Table D.1) that are carried by more than 99% of the advertisement packets (7.9M advertisement records include at least one of those sixteen types). Also, we highlight that the distribution of those AD types varies according to the address type. For instance, the `Complete Local Name` type, respectively found in 36.2% and 82.2% of Public and Random Static addresses, is less observed with Private addresses. This makes sense as those devices are supposed to remain anonymous.

Table D.1 – Presence of AD types in advertising payloads found within $dataset_{Passive}$. Numerical values are fractions of addresses associated with each type.

AD type	Description	Stable		Private	
		Public	Static	Non-res.	Res.
0x01	Flags	87.6	88.6	10.9	77.2
0xff	Manufacturer Specific Data	84.7	51	99.3	93
0x09	Complete Local Name	36.2	82.2	0	1.7
0x03	Complete List of 16-bit Service Class UUIDs	30.1	11.7	0.1	18.7
0x08	Shortened Local Name	25.8	1	0.06	0
0x0a	Tx Power Level	22.1	24.3	0	0.1
0x07	Complete List of 128-bit Service Class UUIDs	10	32.1	0.4	0.5
0x02	Incomplete List of 16-bit Service Class UUIDs	7.9	5.9	0	0
0x19	Appearance	5.3	13.3	0	0.01
0x16	Service Data-16-bit UUID	4.6	54.4	0	17.8
0x12	Slave Connection Interval Range	4.3	4.6	0	0
0x06	Incomplete List of 128-bit Service Class UUIDs	2.6	23.7	0	0.3
0x0d	Class of Device	0.3	0	0	0
0x26	Transport Discovery Data	0.3	0	0	0
0x15	List of 128-bit Service Solicitation UUIDs	0.1	0.1	0	0
0x20	Service Data-32-bit UUID	0.1	0	0	0
Others	Other AD types	1.6	0.2	0	0.003
Overall	Devices that advertise at least one AD structure	99.8	99.2	100	99.9

Appendix E

Additional results for the Bluetooth state change identification attack

Table E.1 – *Apple* iPhone 4 state change identification with AUC values.

From ↓ to →	idle	locked	active	Shazam	BLEScanner	BT inquiry	Wi-Fi Scan
idle	–	0.933	0.942	0.926	0.932	0.953	0.940
locked	0.960	–	0.958	0.955	0.941	0.980	0.942
active	0.940	0.939	–	0.955	0.954	0.964	0.957
Shazam	0.974	0.934	0.936	–	N/A	N/A	N/A
BLEScanner	0.965	0.950	0.972	N/A	–	N/A	N/A
BT inquiry	0.967	0.976	0.960	N/A	N/A	–	N/A
Wi-Fi Scan	0.951	0.922	0.927	N/A	N/A	N/A	–

Table E.2 – *Samsung* Galaxy A3 state change identification with AUC values.

From ↓ to →	idle	locked	active	Shazam	BLEScanner	BT inquiry	Wi-Fi Scan
idle	–	0.933	0.951	0.978	0.970	0.965	0.969
locked	0.922	–	0.958	0.968	0.962	0.961	0.951
active	0.967	0.932	–	0.981	0.951	0.984	0.975
Shazam	0.956	0.959	0.977	–	N/A	N/A	N/A
BLEScanner	0.932	0.913	0.959	N/A	–	N/A	N/A
BT inquiry	0.947	0.977	0.973	N/A	N/A	–	N/A
Wi-Fi Scan	0.985	0.963	0.970	N/A	N/A	N/A	–

Appendix F

Example output of *Himiko*
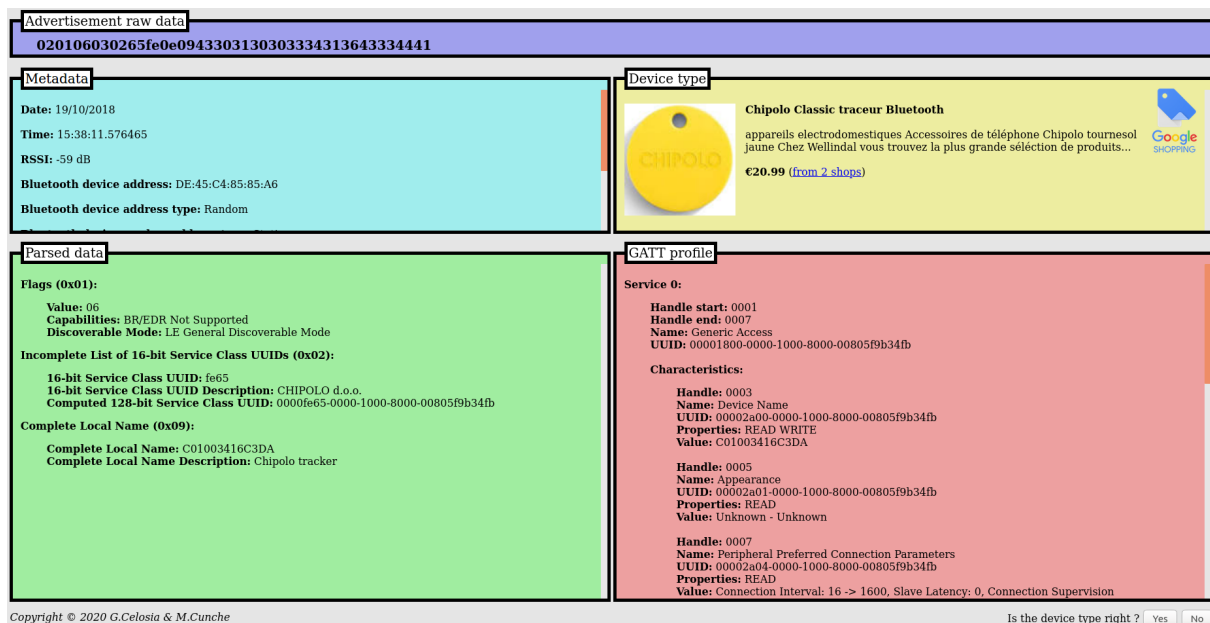
Figure F.1 – Example output of a *Chipolo* Classic BLE keyring device.

Appendix G

Reverse engineered codes of *Apple* BLE Continuity protocols

Table G.1 – Extended list of Magic Switch `Confidence on Wrist` codes.

Confidence on Wrist	Description
0x03	Not on wrist
0x1f	Wrist detection disabled
0x3f	On wrist

Table G.2 – Extended list of Nearby Action `Action Type` codes.

Action Type	Description
0x01	Apple TV Tap-To-Setup
0x04	Mobile Backup
0x05	Watch Setup
0x06	Apple TV Pair
0x07	Internet Relay
0x08	Wi-Fi Password
0x09	iOS Setup
0x0a	Repair
0x0b	Speaker Setup
0x0c	Apple Pay
0x0d	Whole Home Audio Setup
0x0e	Developer Tools Pairing Request
0x0f	Answered Call
0x10	Ended Call
0x11	DD Ping
0x12	DD Pong
0x13	Remote Auto Fill
0x14	Companion Link Prox
0x15	Remote Management
0x16	Remote Auto Fill Pong
0x17	Remote Display

Table G.3 – Extended list of HomeKit `Category` codes.

Category	Description
0x0000	Unknown
0x0100	Other
0x0200	Bridge
0x0300	Fan
0x0400	Garage Door Opener
0x0500	Lightbulb
0x0600	Door Lock
0x0700	Outlet
0x0800	Switch
0x0900	Thermostat
0x0a00	Sensor
0x0b00	Security System
0x0c00	Door
0x0d00	Window
0x0e00	Window Covering
0x0f00	Programmable Switch
0x1000	Range Extender
0x1100	IP Camera
0x1200	Video Doorbell
0x1300	Air Purifier
0x1400	Heater
0x1500	Air Conditioner
0x1600	Humidifier
0x1700	Dehumidifier
0x1c00	Sprinklers
0x1d00	Faucets
0x1e00	Shower Systems

Table G.4 – Extended list of "Hey Siri" `Device Class` codes.

Device Class	Description
0x0002	iPhone
0x0003	iPad
0x0009	MacBook
0x000a	Watch

Table G.5 – Extended list of Nearby Action `Device Class` codes.

Device Class	Description
0x2	iPhone
0x4	iPod
0x6	iPad
0x8	Audio accessory (HomePod)
0xa	Mac
0xc	AppleTV
0xe	Watch

Table G.6 – Extended list of Proximity Pairing `Device Model` codes.

Device Model	Description
0x0220	AirPods
0x0320	Powerbeats3
0x0520	BeatsX
0x0620	Beats Solo3

Table G.7 – Extended list of Nearby Action `Device Model` codes.

Device Model	Description
0x1	D22ish
0x2	SEish
0x3	JEXXish

Table G.8 – Extended list of Proximity Pairing `Device Color` codes.

Device Color	Description
0x00	White
0x01	Black
0x02	Red
0x03	Blue
0x04	Pink
0x05	Gray
0x06	Silver
0x07	Gold
0x08	Rose Gold
0x09	Space Gray
0x0a	Dark Blue
0x0b	Light Blue
0x0c	Yellow

Table G.9 – Extended list of Nearby Action `Device Color` codes.

Device Color	Description
0x00	Unknown
0x01	Black
0x02	White
0x03	Red
0x04	Silver
0x05	Pink
0x06	Blue
0x07	Yellow
0x08	Gold
0x09	Sparrow

Table **G.10** – Extended list of Nearby Action `OS Version` codes.

OS Version	Description
0x09	Version 9
0x0a	Version 10

Table **G.11** – Extended list of Proximity Pairing UTP codes.

UTP	Description
0x01	In Ear
0x02	In Case
0x03	Airplane

Table **G.12** – Extended list of Nearby Info `Activity Level` codes.

Activity Level	Description
0x00	Activity level is not known
0x01	Activity reporting is disabled
0x03	User is idle
0x05	Audio is playing with the screen off
0x07	Screen is on
0x09	Screen on and video playing
0x0a	Watch is on wrist and unlocked
0x0b	Recent user interaction
0x0d	User is driving a vehicle
0x0e	Phone call or FaceTime*

* As reported in [1].

Table **G.13** – Extended list of Tethering Source Presence `Network Type` codes (as reported in [1]).

Network Type	Description
0x01	1xRTT
0x02	GPRS
0x03	EDGE
0x04	3G (EV-DO)
0x05	3G
0x06	4G
0x07	LTE

Appendix H

Siri voice commands

Table H.1 – Complete list of Siri commands used for the dictionary attack on perceptual hashes.

Dictionaries A and B_1 (B_1 is in **bold**)	**Activate Do Not Disturb.**; Add 'tomatoes' to the grocery list.; **Add a reminder.**; **Alarm in 5 hours.**; Alice is my mother.; Best comedy movies ?; **Best horror movies ?**; **Bob is my brother.**; Call 408 555 1212.; **Call Bob.**; **Call Mark.**; Call me sweetheart.; **Call the nearest restaurant.**; Can you recommend a movie ?; Cancel my event with Mark.; Decrease brightness.; FaceTime Audio call to Alice.; Find number of dad.; How are the markets doing ?; **How big is the biggest elephant ?**; How far away is Mars ?; How far away is Tokyo ?.; How humid is it in Paris ?; **How long do cats live ?**; **How many calories in an apple ?**; How many days until Christmas ?; **How many teeth does a cat have ?**; How old is Madonna ?; How old is Peter ?; How tall is Paris Hilton ?; Increase brightness.; **Inform my husband when I'm back home.**; **Inform my wife when I leave my home.**; **Is mom at home ?**; **Learn to pronounce my name.**; **Listen to Alicia Keys.**; **Locate my father.**; Note: 'Susan will be late tonight'.; **Open Instagram.**; **Open Spotify.**; **Play some music.**; Play the rest of this album.; Read my new messages.; Remind me today: call Kevin.; Search Google for pictures of Thor.; Search the web for 'computer'.; **Search the web for 'privacy'.**; **Send a message to Bob.**; **Send a message to Mark.**; **Send an e-mail to Susan.**; Set a timer for 3 minutes.; Show all my photos.; Show me best family movies.; **Show me my notes.**; Show me my photos from London.; Show me my photos of last week.; Show me new e-mail from Peter.; **Show me the latest tweets.**; Show me the nearest cinema.; Show me tweets from Peter.; **Show me videos of Avengers.**; Show my favorite photos.; Show my selfies.; **Square root of 49 ?**; **Susan is my sister.**; Take a video.; **Take me home.**; **Turn on Night Mode.**; **Turn on Wi-Fi.**; What day is in 5 days ?; What day was 2 days ago ?; **What did Dow close at today ?**; **What did Nikkei close at today ?**; What is my altitude ?; **What is the time zone in Miami ?**; **What time is it ?**; What's 7 plus 2 ?; **What's Kevin's address ?**; **What's the capital of France ?**; What's the date ?; **What's the definition of 'robot' ?**; **What's the Nasdaq today ?**; **What's the Nikkei price ?**; What's the temperature outside ?; What's the temperature tonight ?; **What's this song ?**; When am I meeting with Alice ?; When do I meet Bob ?; When is my next appointment ?; When is the sunrise ?; **When is the sunset ?**; When is the Super Bowl ?; Where died Bob Marley ?; **Where is my iPhone ?**; **Where is my next appointment ?**; **Where lives Susan ?**; Which movies are with Tom Hanks ?; **Who does this smartphone belong to ?**; Who is Sean Connery married to ?; **Who sings this ?**
Dictionary B_2	Andrew is my boyfriend.; Any new e-mail from Kevin ?; Call Abraham on speakerphone.; Call me a cab.; Call me king.; Cancel my event with Alice.; Compare Nikkei with Dow.; Deactivate Do Not Disturb.; Delete all alarms.; Delete the reminder 'project'.; Find some movie theaters near my home.; Flip a coin.; How many bones does a dog have ?; How many days until the birthday of dad ?; How many days until year 2020 ?; How small is the smallest dog ?; In which city lives Peter ?; Is 'Airplane mode' enabled ?; Navigate to Susan by car.; Open Facebook.; Open mail.; Open settings.; Pause the timer.; Play me my latest voicemail.; Play the Titanic soundtrack.; Play the trailer for 'Pearl Harbor'.; Play this song from the beginning.; Play top 10 songs from Aya Nakamura.; Read Calendar.; Runtime of Titanic ?; Search Wikipedia for 'dog'.; Show me my photos from Lisbon.; Show me my photos of yesterday.; Show me pictures of Hulk.; Show me the appointments for next month.; Show me the traffic.; Tell me a story.; Text Susan: 'I will be late'; Translate 'cat' from English in Russian.; Turn on Cellular Data.; What is the time at home ?; What's the current dew point ?; What's the Nasdaq price ?; What's the pressure outside ?; What's the visibility outside ?; When died Freddie Mercury ?; When is Kevin's birthday ?; Where is my MacBook ?; Where is the office of Sean ?; Which songs are from Imagine Dragons ?

Appendix I

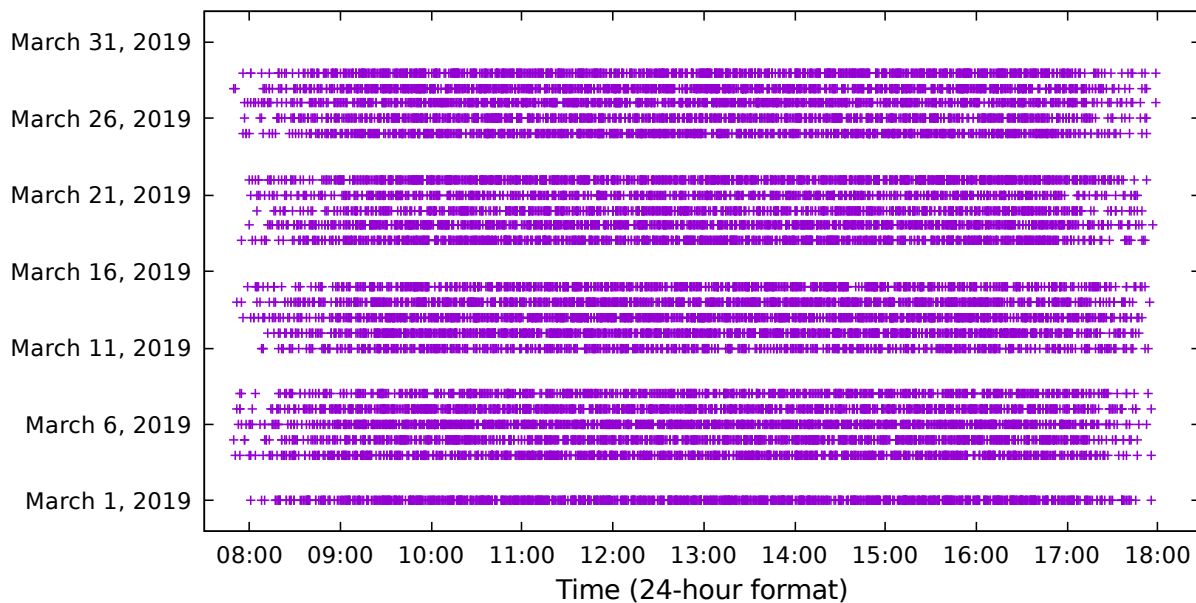# TPMS messages broadcasted by our parked car

Figure I.1 – Representation of a one-month collection of TPMS messages broadcasted by our car parked near our laboratory.

FOLIO ADMINISTRATIF

THÈSE DE L'UNIVERSITÉ DE LYON OPÉRÉE AU SEIN DE L'INSA LYON

Nom : **CELOSIA**   Date de soutenance : **22/09/2020**

Prénom : **Guillaume**

Titre : **Privacy Challenges in Wireless Communications of the Internet of Things**

Nature : **Doctorat**   Numéro d'ordre : **2020LYSEI069**

École doctorale : **InfoMaths**

Spécialité : **Informatique**

Résumé :

Également connue sous le nom d'*Internet des Objets* (IdO), la prolifération des objets connectés offre des opportunités sans précédent aux consommateurs. Des moniteurs d'activité physique aux assistants médicaux, en passant par les appareils électroménagers pour maisons intelligentes, les objets IdO évoluent dans une pléthore de domaines d'application.

Cependant, les avantages qu'ils peuvent apporter à notre société augmentent conjointement avec leurs implications en matière de vie privée. Communiquant continuellement de précieuses informations par le biais de liaisons non filaires telles que le Bluetooth et le Wi-Fi, ces appareils connectés accompagnent leurs propriétaires dans leurs activités. La plupart du temps émises sur des canaux ouverts, et parfois en l'absence de chiffrement, ces informations sont alors facilement accessibles pour tout attaquant passif à portée.

Dans cette thèse, nous explorons deux problèmes de vie privée majeurs résultant de l'expansion de l'IdO et de ses communications sans fil : le *traçage physique* et l'*inférence d'informations utilisateurs*. Sur la base de deux grands ensembles de données composés de signaux radio issus de périphériques Bluetooth/BLE, nous mettons d'abord en échec les fonctionnalités anti-traçage existantes avant de détailler plusieurs applications invasives pour la vie privée. En s'appuyant sur des attaques passives et actives, nous démontrons également que les messages diffusés contiennent des informations en clair allant des caractéristiques techniques des appareils aux données personnelles des utilisateurs telles que des adresses e-mail et numéros de téléphone.

Dans un second temps, nous concevons des contre-mesures pratiques pour résoudre les problèmes de vie privée identifiés. Dans ce sens, nous fournissons des recommandations aux fabricants, et proposons une approche afin de vérifier l'absence de failles dans l'implémentation de leurs protocoles.

Enfin, dans le but d'illustrer davantage les menaces de vie privée enquêtées, nous implémentons deux démonstrateurs. Par conséquent, *Venom* introduit un système de traçage physique visuel et expérimental, tandis qu'*Himiko* propose une interface humaine permettant d'inférer des informations sur les appareils IdO et leurs propriétaires.

Mots-clés : **Vie privée; Réseaux sans fil; Internet des Objets; Objets connectés; Traçage physique; Inférence d'informations utilisateurs; Bluetooth; BLE.**

Laboratoire de recherche : **CITI**

Directeur de thèse : **Daniel LE MÉTAYER**

Président de jury : **Jean-Marie GORCE**

Composition du jury :

Kasper RASMUSSEN (kasper.rasmussen@cs.ox.ac.uk)

Bernard TOURANCHEAU (bernard.tourancheau@univ-grenoble-alpes.fr)

Sonia BEN MOKHTAR (sonia.ben-mokhtar@liris.cnrs.fr)

Jean-Marie GORCE (jean-marie.gorce@insa-lyon.fr)

Vincent NICOMETTE (nicomett@insa-toulouse.fr)

Valérie VIET TRIEM TONG (valerie.viettriemtong@centralesupelec.fr)

Daniel LE MÉTAYER (daniel.le-metayer@inria.fr)

Mathieu CUNCHE (mathieu.cunche@insa-lyon.fr)