Unveiling and Controlling Online Tracking

Jagdish Prasad Achara

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : **Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Jagdish Prasad Achara

Thèse dirigée par **Claude Castelluccia**
et codirigée par **Vincent Roca and Aurélien Francillon**

préparée au sein **d'Équipe Privatics, Inria, Grenoble-Rhone Alpes**
et de **l'École Doctorale MSTII**

Unveiling and Controlling Online Tracking

Thèse soutenue publiquement le 18 octobre 2016,
devant le jury composé de :

Prof. Dr. Bernard Tourancheau
Professor, Université Grenoble Alpes, Président
Prof. Dr. Thorsten Strufe
Professor, TU Dresden, Rapporteur
Dr. Marc-Olivier Killijian
Directeur de Recherche, CNRS-LAAS, Rapporteur
Dr. Nikolaos Laoutaris
Senior Researcher, Telefonica, Examinateur
Dr. Vincent Toubiana
Technologist, CNIL, Examinateur
Dr. Claude Castelluccia
Directeur de Recherche, Inria, Directeur de thèse
Dr. Vincent Roca
Chargé de Recherche, Inria, Co-Directeur de thèse
Dr. Aurélien Francillon
Maître de conférences, Eurecom, Co-Directeur de thèse

Abstract

The main business model of free content and services on the Internet is advertising. As a result, online advertising has become ubiquitous, be it in smartphone apps or on the Web. To deal with the vast demand and supply of ads, various intermediary companies (such as ad platforms and ad exchanges) have come into existence and started to economically compete with each other. As targeted ads are believed to generate better revenues, these companies are massively tracking and profiling users. It has resulted in a privacy nightmare for most Internet users.

This thesis takes a multi-dimensional approach to tackle the problem of privacy in online advertising. First, it encompasses both smartphones and the Web as they are the two major Internet enablers. Second, we act upon all stakeholders: operating system (OS) providers, app developers, advertising and analytics (A&A) companies, webpage publishers, regulators, and users. In fact, by acting upon all stakeholders, we aim to address the privacy problems both at the origin (like the OS providers, A&A companies, and app developers) and at the end user side.

In case of smartphones, the contributions of this thesis address the privacy problems at the origin. It is done by revealing the bad privacy practices employed by different stakeholders (such as A&A companies, app developers, and OS providers) and then, proposing possible countermeasures. To detect privacy leaks on smartphones, we design and implement MobileAppScrutinator, a simple yet efficient dynamic analysis technique for Android and iOS. Using it, we measure and compare user tracking and private data leaks on Android and iOS. This measurement study brings transparency into the system and forces app developers and OS manufactures to take action as their bad privacy practices are revealed to users as well as regulators.

We show how innocuous Android permissions can be exploited by A&A companies to track and profile users. For example, we find that 41% apps (of the 2700 most popular apps on the PlayStore) ask for ACCESS_WIFI_STATE permission even though most of them do not need it for their functionality. In fact, some apps (mostly through the included ad libraries) geo-localise users using this permission without asking for location-specific permissions. Our findings again had a direct impact as Android permission system was later changed and a US-based regulator (FTC) fined one million dollars in 2016 to an ad company (InMobi) for exactly the same issue we reported in 2014.

We also show that the list of installed apps on smartphones is very unique and therefore, there is a significant risk of re-identification of users if a pseudo-anonymised dataset of the list of installed apps is released. As this list is available to be accessed by Android apps without requiring any user permission and as A&A companies implicitly know a subset of installed apps in which they are present, we show that Android poses its users a serious privacy threat. We reported the issue to Google and hope to address the problem at the origin.

In case of the Web, the contributions of this thesis are to provide end users with tools to have more transparency and control over online tracking and advertising. In terms of transparency, these tools allow users to know what profile trackers and advertisers know about them and why an ad is delivered to them. With our proposed tools, users can also

exercise fine-grained control like choosing the categories of web pages where they do not want to be tracked and the kinds of ads they do not want to receive. We believe that providing users with fine-grained control allows them to take social and economic benefits of online advertising while technically enforcing their privacy preferences at the same time. Moreover, such control can also support the current ad-supported economic model of the Web.

Keywords: Privacy, Web, Smartphones

Résumé

Cette thèse suit une approche multi-dimensionnelle pour le problème du respect de la vie privée dans l'Internet que nous connaissons aujourd'hui. Nous considérons pour cela les deux principaux modes d'accès, les smartphones et le Web. De plus nous considérons les différents acteurs (fabricants de systèmes d'exploitation, développeurs d'applications, régies publicitaires, éditeurs de contenus Web, autorités de régulation et utilisateurs). En fait, en agissant sur les acteurs, nous essayons de corriger les problèmes de respect de la vie privée à la fois à la source (par exemple au niveau des fabricants de systèmes d'exploitation, des régies publicitaires et des développeurs d'applications) et au niveau des utilisateurs.

Dans le cas des smartphones, cette thèse contribue à corriger les problèmes à leur source, en identifiant et en révélant les mauvaises pratiques pour ensuite proposer des contre-mesures. Dans le cas du Web, nous fournissons aux utilisateurs finaux des outils permettant transparence et contrôle.

Nous avons conçu et réalisé MobileAppScrutinator comme un outil à la fois simple mais efficace d'analyse dynamique pour Android et iOS. Avec l'aide de MobileAppScrutinator, nous avons mesuré et comparé le traçage des utilisateurs et les fuites de données personnelles des 140 applications les plus populaires à la fois sur Android et iOS. Cette étude apporte de la transparence au système et force les développeurs d'applications et fabricants de systèmes d'exploitation à prendre des mesures car leurs mauvaises pratiques sont exposées au public et aux autorités de régulation.

Nous avons montré qu'une permission d'aspect anodin pouvait être utilisée pour collecter des données personnelles et que 41% des 2700 applications les plus populaires demandaient cette permission. Nous avons trouvé que certaines d'entre elles (essentiellement au travers de bibliothèques publicitaires tierces incluses) géolocalisent les utilisateurs sans passer par la permission de collecte de géolocalisation. Nos résultats ont ici encore eu un impact direct puisque le système d'autorisation Android est modifiée et le régulateur américain a infligé à la société publicitaire une amende d'environ un million de dollars.

Enfin, nous avons montré que la liste des applications installées sur un smartphone est unique. Par conséquent il y a un fort risque de ré-identification des utilisateurs si un jeu de données pseudo-anonymisées avec des listes d'applications installées venait à être rendu public. Comme la liste des applications installées sur Android est accessible aux applications sans avoir à obtenir de permission dédiée, Android expose là aussi l'utilisateur à de sérieuses menaces en termes de vie privée.

Nous avons ensuite fourni aux utilisateur du Web des outils qui apportent transparence et contrôle vis à vis des techniques de traçage et de ciblage publicitaire. En termes de transparence, les utilisateurs peuvent savoir pourquoi une publicité leur est délivrée, quel profil les sociétés de traçage et régies publicitaires connaissent d'eux. Nos outils permettent également aux utilisateurs d'avoir un contrôle précis. Par exemple, les utilisateurs peuvent choisir les catégories de pages où ils ne désirent pas être tracés ainsi que le type de publicités qu'ils ne veulent pas recevoir. En fait, fournir un tel contrôle est essentiel pour permettre au modèle économique du Web, basé sur la publicité, de perdurer.

Mots-clés: Vie privée, Web, Smartphones

List of Publications During the PhD

International Conferences, Workshops and Journals

Jagdish Prasad Achara, Javier Parra-Arnau, and Claude Castelluccia. MyTracking-Choices: Pacifying the Ad-Block War by Enforcing User Privacy Preferences. *The 15th Annual Workshop on the Economics of Information Security (WEIS)*, 2016

Jagdish Prasad Achara, Gergely Acs, and Claude Castelluccia. On the Unicity of Smartphone Applications. In *The 14th ACM Workshop on Privacy in the Electronic Society (ACM WPES)*, 2015

Gergely Acs, Jagdish Prasad Achara, and Claude Castelluccia. Probabilistic Km-anonymity: Efficient Anonymization of Large Set-valued Datasets. In *IEEE International Conference on Big Data (IEEE Big Data)*, 2015

Célestin Matte, Jagdish Prasad Achara, and Mathieu Cunche. Device-to-identity Linking Attack Using Targeted Wi-fi Geolocation Spoofing. In *The 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks (ACM WiSec)*, 2015

Jagdish Prasad Achara, Mathieu Cunche, Vincent Roca, and Aurélien Francillon. Short Paper: WifiLeaks: Underestimated Privacy Implications of the Access_Wifi_State Android Permission. In *The 7th ACM Conference on Security and Privacy in Wireless & Mobile Networks (ACM WiSec)*, 2014

Jagdish Prasad Achara, James-Douglass Lefruit, Vincent Roca, and Claude Castelluccia. Detecting privacy leaks in the RATP App: how we proceeded and what we found. *Journal of Computer Virology and Hacking Techniques (Springer JCVHT)*, 2014

Under Submission

Jagdish Prasad Achara, Vincent Roca, Claude Castelluccia, and Aurélien Francillon. MobileAppScrutinator: A Simple yet Efficient Dynamic Analysis Approach for Detecting Privacy Leaks across Mobile OSs. *Submitted to The 32nd Annual Computer Security Applications Conference (ACSAC)*, 2016

Javier Parra-Arnau, Jagdish Prasad Achara, and Claude Castelluccia. MyAdChoices: Bringing Transparency and Control to Online Advertising. *Submitted to The ACM Transactions on the Web (ACM TWEB)*, 2016

Acknowledgements

Firstly, I sincerely thank my advisers, Claude Castelluccia, Vincent Roca, and Aurélien Francillon. Without their continuous support and motivation, this work would not have been possible. They showed great patience while responding to my countless questions and were always available when I needed them. They provided me with a good balance between freedom of work and guidance. I could not have imagined having better and friendly advisors to complete this journey.

I'm sincerely thankful to all other co-authors in alphabatical order: Celestin, Gergely, James, Javier, and Mathieu. I have learnt a lot while working with them and it has indeed been a pleasant experience. If I can say one thing that I learnt from them, I would say: Gergely introduced me to the beautiful world of zeal towards learning new things (it gives me immense pleasure to learn new things now), I hope that one day I can have the same patience and organisation capabilities as Javier, and finally, I can enjoy the research as much as Mathieu by being creative and doing one thing at a time.

I would like to express my sincere gratitude to Prof. Dr. Thorsten Strufe and Dr. Marc-Olivier Killijian for reviewing this dissertation. I would also like to thank other jury members: Prof. Dr. Bernard Tourancheau from UGA, Dr. Nikolaos Laoutaris from TID, Barcelona, and Dr. Vincent Toubiana from CNIL. It is a great honor for me that they accepted to be part of my jury.

I cannot restrain myself from saying many thanks to Amrit, Levent, and Pierre. They have been very helpful to me: giving practical advices on just about anything, helping me in administrative tasks, proof-reading my papers, and many more. In the cafeteria, we had interested discussions. Merci beaucoup à vous tous. My thanks also go to others whose names I forgot to mention here.

Last but not the least, I do not have enough words to say thanks to my wife. I cannot forget her sacrifice on many occasions: when I had to work on weekends, late night deadlines, mood switches, burnouts, not responding to her calls, my absence when she needed me the most, and many more. I will always be grateful to my parents: they have been amazing and let me choose what I wanted to do in my life. Their continuous love, support, and encouragement is the key to my success. No matter how hard I would have tried, it would have been almost impossible for me to get the PhD title without the support of my family.

The work presented in this thesis was supported in part by Cappris (Collaborative Action on the Protection of Privacy Rights in the Information Society), an Inria Project Lab.

Contents

Chapter 1

Introduction

Contents

1.1 Context of this work

The context of this work is investigating and enhancing privacy in advertising on the Internet. Below we give a necessary overview of online advertising and privacy on the Internet.

1.1.1 Online Advertising

During the last two decades, the Internet has been gradually integrated into people's daily lives, enabling new forms of communication. The so-called network of networks has become an essential communication channel not only among people, but also among businesses and their customers.

Breathing new life into traditional business activities is one of the Internet's most relevant influences. The Internet has led to key business changes embracing the whole value chain in almost all sectors and companies. These changes have had an impact on how products are sold and also more importantly, on how companies approach customers in a personalised manner, taking into account their unique preferences.

The industry of advertising is a clear example of the transformation driven by the ever-growing sophistication of Internet technologies. In the past, ads were served directly following a one-size-fits-all approach. But due to the gradual introduction of intermediary companies with extensive capabilities to track users, Internet advertising has become increasingly personalised and pervasive.

The ability of the online marketing industry to track and profile users' Web-browsing activity is therefore what enables more effective, tailored advertising services. The intrusiveness of these practices and the increasing invasiveness of digital advertising, however,

have raised serious concerns regarding user privacy and usability. According to recent surveys, two out of three Internet users are worried about the fact that their online behaviour is being scrutinised without their knowledge and consent [177]. Numerous studies in this same line reflect the growing level of ubiquity and abuse of advertising, which is perceived by users as a significant degradation of their browsing experience [39, 154, 171].

1.1.2 Privacy

What is privacy and why is it hard to achieve? Even though privacy can be defined in various ways, in this thesis, we define it as follows: if an individual can control the information[1] others know about her, she can be said to have privacy. As we see, the definition of privacy is simple and straightforward. But privacy is hard to achieve in practice due to the inability of most users to exert their control over their information. There are multiple reasons for this inability. One reason is that inferences or consequences are not always clear in advance for most users. Another reason why individuals cannot always exert control is the lack of transparency of practices. Finally, lack of technical expertise can be one more reason why people cannot exert the control over their personal information. As most users are not trained to use computers, they do not know how to use online services in a private manner and how to have a proper control over their data. For example, installing add-ons or extensions in a browser in order to block web trackers requires some know-how that is not shared by all users.

Why is privacy important? Privacy is essential for individuals and citizens. It is a right granted to individuals that underpins the freedoms of expression, association, and assembly, all of which are needed for a free, democratic society. In other words, loss of privacy leads to loss of freedom. For example, in case of the Internet, freedom of expression is threatened if our Internet usage is surveilled, freedom of association is threatened if our online communications are surveilled, and lastly, freedom of assembly is threatened if our participation is tracked [57].

1.2 Problem statement

The problem of privacy on the Internet today is a fallout of rapid growth in the technology. The technology developed so fast that privacy could not keep the pace with it. Below we discuss the main reasons why the Internet today is suffering from privacy issues. This thesis aims to tackle these reasons:

- **Lack of transparency.** As already said, on the Internet, users benefit from ad-supported content and services. Marketers track and profile users because targeted ads are believed to bring more revenues. This can be a good trade-off if users are aware of ongoing data collection. But today this is not the case. Most of the time a user cannot answer questions such as: what data is being collected, for what purpose, and how is collected data processed and stored by the data collector? For example, users do not know what apps installed on their smartphones are doing [102, 103, 33, 187, 207], how and by whom they are being tracked over the Web [105], and what data OS provider companies (such as Google and Apple) are collecting [89]. If these important questions remain un-answered, users cannot take appropriate decisions. In a nutshell, the lack of transparency is at the center of the problem because it prevents users from exerting control over their information.

[1] Here it is to be noted that the amount of information may vary from one person to another.

- **Lack of control.** While users browse the Web on their computers, they leave many digital traces. Moreover, new devices (such as smartphones or smartwatches) are being introduced in the lives of people and they generate new kinds of data about users. This data includes various information from in-built sensors, call logs, and other meta-data. Users do not have sufficient tools today to control this data and hence, their information.

- **Difficulty of understanding privacy implications.** Apart from the problem of transparency, the Internet privacy today suffers from lack of understanding of privacy implications. In other words, individuals may sometimes share additional information unwillingly because they cannot predict what information can be derived from increasingly sophisticated forms of data and services on the Internet [61, 208]. Users sometimes share data because they do not know that some sensitive information (which they would not like to share if they knew) can be derived. For instance, New York Taxi and Limousine Commission (NYC TLC) released pseudo-anonymised trip records for taxis in New-York [195]. Since the database includes passenger pick-up and drop-off dates/times, muslim taxi drivers can easily be identified by looking at the inactivity periods of the drivers [77]. As privacy implications of data are sometimes discovered after the data is shared, users are not always in a position to exert the control over the information they are willing to share with others.

- **Lack of viable solutions/tools.** Another problem with tools provided to end users today is that they do not take all the aspects into considerations and therefore are not always viable. For example, in case of online advertising on the Web, various tools (such as ad-blockers) [11, 2, 21] are proposed to counter the privacy problems but they do not consider their economic impacts. Therefore, these tools cannot be considered as viable solutions: they create more problems in the long-term than they solve in the mid-term [42].

1.3 Contributions

As privacy is difficult to achieve on the Internet due to aforementioned reasons, various attempts have been made by researchers, technologists, privacy enthusiasts, and regulators to enhance it. In particular we think that the following three directions are of particular importance:

- increase transparency, and therefore the end-user awareness, towards how data is being collected, shared and used while users are connected to the Internet;

- better understand and expose the privacy implications of data we share and services we use;

- give control back to users over their data.

The contributions of this thesis are along these lines. In terms of giving more control to the users, we advocate for making it as fine-grained as possible.

As online advertising is present both in smartphone apps and the Web, this thesis encompasses the two. Moreover, we act upon all stakeholders to enhance privacy: operating system (OS) providers, app developers, advertising and analytics (A&A) companies, webpage publishers, regulators, and users. This is because we believe that a multi-dimensional approach can tackle the privacy problems in a faster and more efficient manner.

To better understand the global contributions of this thesis, we look at the bigger picture, with all stakeholders. We attempt to tackle the privacy problems either at the

Ecosystem \ Approach	**Solving privacy problems at the source/origin**	**Empowering users**
Smartphones	1) Measuring and revealing bad privacy practices 2) Proposing countermeasures	Transparency and fine-grained control
Web	1) Measuring and revealing bad privacy practices 2) Proposing countermeasures	Transparency and fine-grained control

Table 1.1: The table gives the global picture of efforts needed to be done to enhance privacy on the Internet. The gray colored cells are the ones on which this thesis contributes: in case of smartphones, we address the privacy problems at the origin whereas in case of the Web, at the user side.

origin/source (such as OS providers, app developers, website publishers) or at the end user side. To tackle the problems at the origin, we detect and measure the bad privacy practices and then, notify the results to the public, regulators, and other stakeholders. Then, to address the problems at the end user side, we provide more transparency and fine-grained control to users.

In case of smartphones, the contributions of this thesis are to tackle the privacy problems at the origin by measuring and revealing the bad privacy practices of different stakeholders. This is because it is hard to develop tools for end users on smartphones as they are relatively closed in nature when compared to web browsers. To develop tools for end users that provides them with transparency and control over other apps on smartphones, one typically has to either get the root privileges on the phone or change the firmware from scratch. Both approaches are not good as they prevent large scale deployment of the tool and pose security risks to the users. However, in case of the Web browsers, we do not face the same issues and hence, we provide end users with tools.

In short, the global picture of what we need to do to enhance privacy is summarised in Table 1.1. Those cells where this thesis makes contributions are colored in gray. Next we give a detailed account of contributions in each of these cases.

1.3.1 Unveiling online tracking on smartphones

On smartphones, we address the privacy problems at the origin. To detect privacy leaks, we designed and implemented MobileAppScrutinator, a simple yet efficient dynamic analysis technique for Android and iOS, the two most popular operating systems for Smartphones. With the help of MobileAppScrutinator, we measured and compared user tracking and private data leaks by the 140 most popular Android and iOS apps. This measurement study brought transparency into the system.

We also revealed a flaw in the Android permission system. We showed that an innocuously looking permission, called ACCESS_WIFI_STATE, can be used to gather user private data and that 41% out of the 2700 most popular apps (100 top apps in each 27 categories) required this permission. By statically and dynamically analysing these apps, we found that some apps (mostly through the included ad libraries) are geo-localising users without asking for location-specific permissions. Our findings again had a direct impact as Android now protects the most privacy-critical API (getScanResults()) with location permissions so that apps can no more exploit this innocuous permission to geo-localise users. Moreover, the exact issue we reported in our study in 2014, i.e., geo-localising users without their permission, led Federal Trade Commission (FTC) to fine InMobi for an amount of almost one million dollars in 2016.

Lastly, we showed that the list of installed apps on smartphones is unique and therefore, there is a huge risk of re-identification of users if a pseudo-anonymised dataset with the list of installed apps is released. As the list is available to be accessed by Android apps without requiring a user permission, and as A&A companies implicitly know a subset of installed apps in which they are present, we proved that Android poses its users a serious privacy threat. We reported our findings to the Android security team in the hope that this issue can be fixed as soon as possible.

1.3.2 Unveiling and controlling tracking on the Web

In a second step, to enhance privacy on the Web, we also designed and provided end users with two tools: *MyAdChoices* and *MyTrackingChoices*. These tools were aimed at bringing transparency and control to users over online tracking and advertising.

MyAdChoices is a tool that aims at bringing transparency to online advertising, so that users can make an informed and equitable decision regarding ad blocking. When an ad is delivered to a user, the tool decides and lets the user know if that ad is retargeted, or is delivered based on the past browsing history of the user. It is implemented as a Web-browser extension and enables users to exert fine-grained control over advertising, thus providing them with certain guarantees in terms of privacy and browsing experience, while preserving the Internet economic model. Experimental results in a real environment demonstrate the suitability and feasibility of this tool, and provide preliminary findings on behavioural targeting from real user browsing profiles.

MyTrackingChoices provides users with fine-grained control over tracking. It allows users to select the categories of web pages that are privacy-sensitive to them (for example, health, religion) and block network connections of third-party domains (and thus, ads delivered through third-party domains) present only on such web pages. Since users will continue receiving ads on web pages belonging to non-sensitive categories, our approach essentially provides a trade-off between privacy and economy. Basically, *MyTrackingChoices* helps users to control their profile collected by different third-parties, like advertising companies, data brokers, or insurance companies, without totally preventing ads. We believe that our proposal can provide an appropriate level of privacy while simultaneously supporting the ad-based economic model of the Web.

1.4 Organisation of the thesis

This thesis is divided into two parts and is a compilation of papers with several co-authors. I was the main author for all these papers except [175]. My contributions in [175] were practical design and implementation of the whole system.

The first part of the thesis is about the work we have done to unveil the privacy problems in case of smartphones. It includes chapter 2 that discusses the MobileAppScrutinator tool, chapter 3 that discusses the privacy implications of ACCESS_WIFI_STATE android permission, and chapter 4 that discusses the re-identification risk posed by unicity of apps installed on smartphones.

The second part is about addressing privacy problems at the end user side on the Web. It includes chapter 5 that discusses the MyAdChoices tool, and chapter 6 that discusses the MyTrackingChoices tool. Finally Chapter 7 concludes this work and gives some possible future directions.

Part I

Unveiling Online Tracking on Smartphones

Preface

Approach / Ecosystem	Solving privacy problems at the source	Empowering users
Smartphones	WSJ study [33], PiOS [102], TaintDroid [102], Han et al. [123]	HayStack App [178]

Table 1.2: The table gives some notable projects taking two approaches: solving privacy problems at the source and empowering users. This list of projects is not exhaustive. In this thesis, we take the first approach, i.e., solving privacy problems at the source/origin. The related projects taking this approach are colored in gray.

Mobile devices have become ubiquitous and are a crucial part of our lives today. This is mainly due to the wide variety of functionalities they provide beyond telephony. They are equipped with a lot of different sensors, e.g., GPS navigation unit, Camera, Accelerometer. They handle a lot of private data such as our email communications, location. As useful as they are, they have also became a serious threat to user privacy. As very accurate profiles can be created using the vast amount of data available on them, Advertising and Analytics (A&A) companies have therefore shifted their focus from traditional desktop computers and browsers to applications running on mobile devices.

To deliver ads in smartphone apps, app developers generally include code from A&A companies in their apps. This code is typically responsible for both ad delivery and collecting personal data about users to create a profile of users' interests. As the code from A&A companies run with same privileges as the app, it is difficult for users to know if their personal data is accessed by the app or the A&A companies or both. From this point of view, smartphone apps generally remain a blackbox for users.

The goal of this part of the thesis is to understand online tracking on smartphones. Specifically, we want to study how A&A companies are tracking and profiling users. There can be two approaches to do this task: scrutinise apps in the lab or give tools to the end users. In this thesis, we take the first approach because monitoring other apps on smartphones typically requires elevated (root) privileges and thus, makes the tool/solution less attractive. Indeed, asking users to get elevated privileges requires them to jailbreak the phone or modify the firmware. Evidently, this is not a scalable solution and poses security risks to the users.

Table 1.2 presents the related projects taking these two approaches. It is worth emphasising that this list of related work is not exhaustive. These are the main projects which tried to tackle this problem of privacy on smartphones using these two approaches. The goal of all these projects was to bring transparency into the world of smartphone apps. They showed the data that is being collected by different smartphone apps, how frequently it is being collected and by whom. Below we give more details about these projects.

One of the notable projects in the first approach is WSJ study [33]. They analysed the data collected from 101 popular Android and iOS apps [33]. In this study, they found

that both A&A companies and app developers are collecting unique identifiers as well as location and other personal information about users. They also studied the privacy policies of these apps and found that there were a lot of discrepancies between what these apps were actually doing and what their privacy policy is saying.

Han et al. also took the first approach and shedded some light on third-party tracking on Android using TaintDroid [123]. Also, all other static and dynamic analysis tools proposed in the literature, most notably, PiOS [102] and TaintDroid [103], analysed some applications and presented a number of applications leaking user PII.

HayStack app [178] takes the second approach and tries to provide users with transparency on smartphone apps. It is different from studies mentioned above as the app can be installed by smartphones users without requiring elevated privileges or modifying the firmware. It lets users find the trackers present in their apps, lets them know what data the apps are collecting, and who is collecting that data. It captures the leakage of data both over clear-text and SSL. The app leverages standard VPN API and therefore, does not require root privileges.

In the end, we note that the work done in this thesis is in no way exhaustive to solve the problem of user tracking on smartphones. The problem still persists and we need to continue working on providing all stakeholders with more transparency and control.

Chapter 2

Measuring Private Data Leaks using MobileAppScrutinator

Contents

2.1 Introduction

Smartphones no longer involve only the user and the communication service (GSM/CDMA) provider. The revolution of the AppStore model for application distribution brings a large number of new actors. In the literature, service providers to whom the user directly interacts with are considered as first-party, the user being the second-party. However, there are many additional actors whose services are not directly used by the end user, and whose presence is not obvious to most users, are called the third-parties. Third parties

are, for example, Advertisers and Analytics (A&A) companies, application performance monitors, crash reporters, or push notification senders. The situation has become even more complex with the development of new advertisement models like Mobile Ad Networks or Ad exchange networks for real-time bidding (RTB).

Depending on the service provided, a user may accept to exchange data with a first-party, in general following legal terms and conditions upon which they mutually agree. However, the data collection by third-parties without explicit user consent is more problematic. Due to economic reasons, the A&A companies are the dominant third parties and the most privacy intrusive. Indeed, in order to increase their revenue, advertisers want to send personalized Ads to the user. Therefore A&A companies are incited to collect as much information as possible to better profile user's interests and behavior. In order to achieve this goal, they need a way to identify the smartphone/user via an identifier that can uniquely be associated with a smartphone/user. This whole process of data collection is called **"third-party smartphone tracking"** or simply **"tracking"** and the process of showing user-specific Ads based on user profile is called **"targeted advertising"**.

Smartphone tracking and targeted-advertising are acceptable if the user is aware of it and if he agrees to receive targeted Ads based on his personal interests. Some users could also find the presence of third-parties on the smartphone beneficial. However, problems arise when A&A companies collect Personally-Identifiable Information (PII) without users' knowledge. In fact, Ad libraries sometimes also include APIs through which an application can deliberately leak user PII [80]. This creates serious privacy risks for users. With the rapidly growing number of smartphones, people are increasingly exposed to such risks. Moreover, a smartphone is particularly intrusive, revealing all user movements as it is equipped with many sensors, and it stores a plethora of information either generated by these sensors, by the telephony services (calls and SMS), or by the user himself (e.g., calendar events and reminders). Finally, various scandals in the past (e.g.,[33, 31]) make it difficult to trust all these actors present on smartphones.

Motivation As "Tracking these trackers" has become a necessity, various tools have been developed in the past. These tools are based on either static analysis or dynamic analysis or interception of network traffic. Even though static analysis techniques scale well, they generally fail on obfuscated applications and therefore, are not suited to accurately detect and measure the ongoing tracking. Similarly, network interception is often unable to handle SSL traffic. Dynamic analysis techniques for detection and measurement of PII leaks are available on Android but not on iOS. As we lack suitable dynamic analysis tools readily available on *both* Android and iOS, there is no measurement study in the literature which provides concrete evidence of ongoing tracking as well as the comparision across mobile OSs.

Contributions The contributions of this chapter are threefold:

1. We present our dynamic analysis platform, *MobileAppScrutinator*, which detects and measure the ongoing tracking on Android and iOS. It is the first dynamic analysis platform for iOS to detect private data leakage.

2. We detect private information leaks by applications over the Internet, when they leak it in clear-text or over SSL. MobileScrutinator detects the PII leakage even if the App obfuscates the data (by hashing or encrypting) before sending over the network in clear-text or through SSL. Detection of modified PII is a key for reliable measurements as some identifiers (e.g., WiFi MAC address, AndroidID, IMEI) are often modified before being sent.

3. We test 140 popular applications with *MobileAppScrutinator*, on both Android and iOS, and report our findings with a comparision of ongoing tracking on these two platforms.

4. Finally, we discuss the effectiveness of privacy safeguards available on both Android and iOS. We show that neither Android nor iOS privacy safeguards in their present state are completely satisfying.

2.2 Related work

Our work can be compared with existing works on two axes: 1) Tracking measurement technology/tools and 2) Measurement of PII leakage. Below we discuss and compare our work with some most representative works along these two axes.

2.2.1 Tracking measurement technology

Tools to measure the ongoing tracking might be based on either interception of generated network traffic, or static analysis of the application code, or dynamic analysis of applications.

Interception of generated network traffic This approach is based on snooping the network data using Man-In-The-Middle (MITM) proxy. For example, MobileScope [16], based on MITM proxy, was used in WSJ study [33] to investigate the top 100 applications on both Android and iOS. However, this technique cannot be used to intercept the SSL traffic which seriously limits the effectiveness of this approach; as we see in Sections 2.5 and 2.6 that almost half of PII leakage is through SSL. Additionally, MITM approach will not be able to detect reliably the leakage of PII generated by the system (values not known to the user and therefore, could not be searched in the network traffic). This includes different PII, for example, unique IDs generated and shared by applications and user location. Also, MITM based approach would fail in cases where user PII is modified (e.g., hashed or encrypted) before being sent (and, as we will show, this is a rather common practice). Finally, being a network packet analysis approach, it is not always easy or feasible to identify the application having generated the monitored traffic, which makes the (manual) analysis even more complex. MobileAppScrutinator, in contrast, makes analysis directly at the operating system level, and thus does not suffer from such limitations.

Static analysis Past works (PiOS [102] on iOS, FlowDroid [71], ScanDroid [113], CHEX [149], AndroidLeaks [117], SymDroid [129], ScanDal [133] and AppIntent [205] on Android) are based on static analysis to detecte a flow of data from a PII source to a network sink. These works can be classified in two categories: 1) static tainting-based (e.g., FlowDroid [71], ScanDroid [113], CHEX [149], AndroidLeaks [117]) and 2) symbolic execution based (e.g., SymDroid [129], ScanDal [133] and AppIntent [205]). Among the ones based on symbolic execution, SymDroid [129] designs a symbolic executor based on their simple version of Dalvik VM, i.e., micro-dalvik. Similarly, ScanDal [133] designs an intermediate language, called Dalvik Core, and collects all the program states during the execution of the program for all inputs. Considering the Android's special event-driven paradigm, AppIntent [205] proposes a more efficient event-space constraint guided symbolic execution. On iOS, PiOS was designed for binaries compiled with GCC/G++ compiler and since then, Apple switched to LLVM compiler. Therefore, PiOS needs to be adapted to support the analysis of binaries compiled with LLVM. Furthermore, PiOS is not available publicly, so one needs to build it from scratch to use it to detect and measure

the ongoing tracking. In general, static analysis techniques do scale well but they lack dynamic information tracking and therefore, lead to false negatives.

Dynamic analysis TaintDroid [103] and PMP [66] are based on dynamic analysis on Android and iOS respectively. On Android, TaintDroid is a dynamic taint-based technique to detect and measure private data leakage. However, TaintDroid has its own limitations: 1) taint-based tracking can be easily circumvented using indirect information flows [184] 2) requires to make a trade-off between false positives and false negatives ([103] did not taint IMSI due to false positives) 3) misses native code (both for taint propagation and as a source of information). On iOS, PMP [66] is a dynamic/runtime tool that offers the functionality of choosing access to what information a user is willing to share with a particular application. As iOS's own privacy control feature provides the same functionality, PMP [66] is an enhancement in terms of the number of different types of private data considered. In fact, PMP fails to notify users if the accessed information is being sent to a remote server or not. As existing dynamic analysis tools on iOS are not sufficient to measure the ongoing tracking on iOS, one possibility could have been a taint-based dynamic analysis technique. However, this is not possible to do on iOS device because the code of applications is native (C, C++ and Objective-C). Propagating taint would require to emulate native code, which involves serious changes to the system and would have a significant performance penalty. Therefore, we opted for a dynamic analysis approach, described in the next section, which can be used on both Android and iOS. To measure the effectiveness of MobileAppScrutinator on Android with TaintDroid, we perform tests on identical set of applications using TaintDroid and MobileAppScrutinator. We found that MobileAppScrutinator could detect a lot more PII leakage than TaintDroid.

2.2.2 Measurement of PII leakage

To best of our knowledge, no previous work provides a complete picture of tracking on Android and iOS. Web-browser tracking has been thoroughly studied [156, 181], but it is not the case with smartphone tracking. We are first to provide detailed analysis and measurement data for both Android and iOS. [123] sheds some light on third-party tracking being taken place on Android using TaintDroid but is not as comprehensive as ours. Also, all other static and dynamic analysis tools proposed in the literature, for example, PiOS [102] and TaintDroid [103], analyzed some applications and presented a number of applications leaking user PII, but none of them presented a complete analysis as we do in Section 2.5 and 2.6. Also, as tracking technologies rapidly change with OS revisions, it is crucial to have up-to-date tools and a recent picture of tracking technology. Furthermore, we also consider the remote servers where PII is sent and attempted to distinguish them among first and third-parties, with the available information.

2.3 MobileAppScrutinator

Design choices From tracking detection and measurement point of view, it is ideal to analyse what applications are doing at operating system level. However, we want to have a system that does not require too intrusive modifications of the OSs and does not have too many false positives (unlike dynamic tainting-based techniques). The system should work well with non-malicious application on real devices so that it can be used by anyone. Thus the design of MobileAppScrutinator starts with a simple approach: *intercepting the source, sink and data manipulation system APIs.* As the same approach is applied to both Android and iOS, it enables us to compare the tracking across mobile OSs. Even though MobileAppScrutinator is based on this same simple approach on both Android

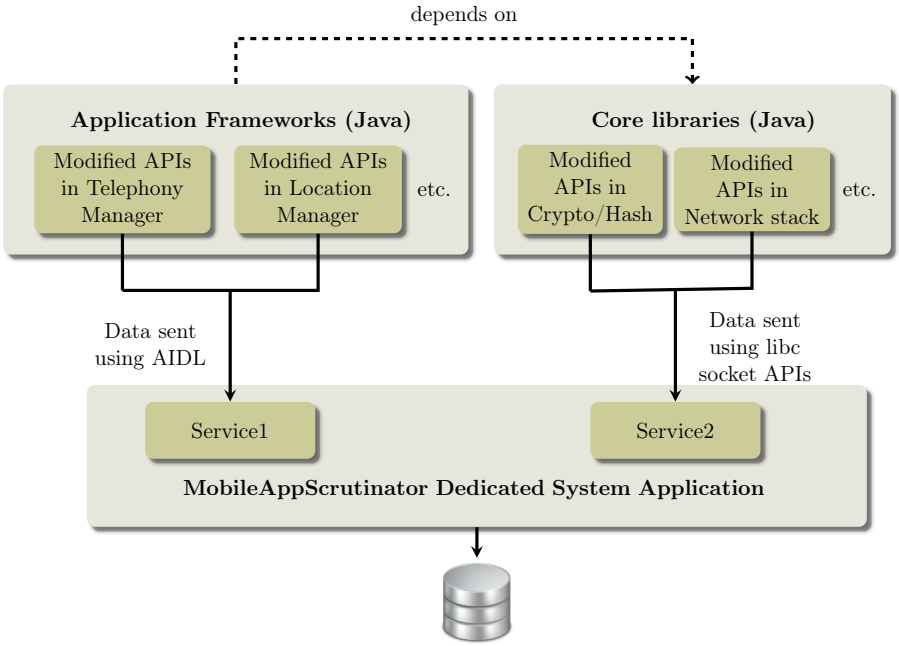


Figure 2.1: MobileAppScrutinator Android implementation.

and iOS, i.e., the basic governing philosophy remains the same, its implementation *differs significantly* due to the differences of these OSs.

Overall architecture As developer APIs are public on both Android and iOS, we are able to identify all source and sink methods (i.e., methods related to access or modification of private data along with network operations either in clear-text or encrypted). MobileAppScrutinator hooks these APIs and includes extra code to these APIs. This added extra code collects various information from the application environment. Specifically, it collects information about PII being accessed, modified, or transmitted by an application along with the information about that application. Any access, or modification, or transmission of PII corresponds to an event and is stored locally in an SQLite database. This database is later analyzed automatically to detect and measure privacy leaks.

In the next two subsections, we give details of how MobileAppScrutinator is implemented on Android and iOS.

2.3.1 Implementation on Android

As the Android source (from Android Open Source Project (AOSP)) is publicly available, MobileAppScrutinator directly modifies source code of various APIs in Java frameworks. This modified source is compiled and a new system image is generated. We develop and add a system application to this new system image. This system application runs two Android services that are responsible for receiving data from different sources. The App is also responsible for storage of data in a local SQLite database.

As described earlier, our added extra code must send data to MobileAppScrutinator system app. To do so, we use two methods: AIDL[1] and socket APIs of libc library. Our extra coded added in Android application frameworks APIs uses AIDL to send data to MobileAppScrutinator system app whereas socket APIs of libc library are used in modified core Java frameworks. Here it is to be noted that Android application frameworks are written utilizing core Java frameworks, i.e., during compilation of Android source, core

[1] `http://developer.android.com/guide/components/aidl.html`

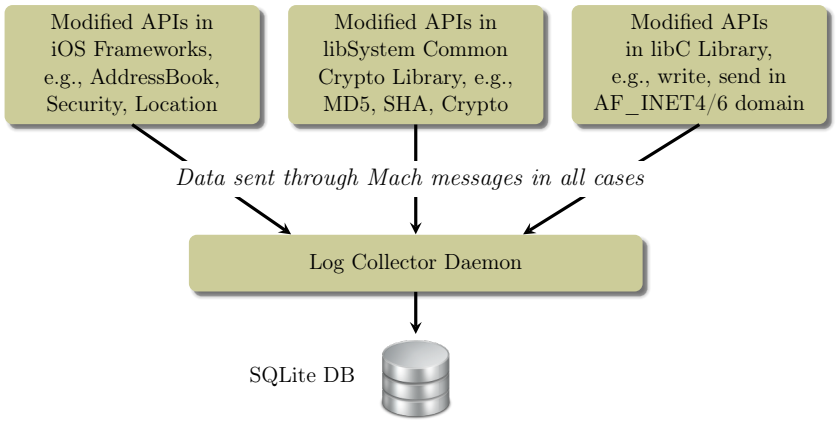


Figure 2.2: Overall picture of MobileAppScrutinator iOS implementation.

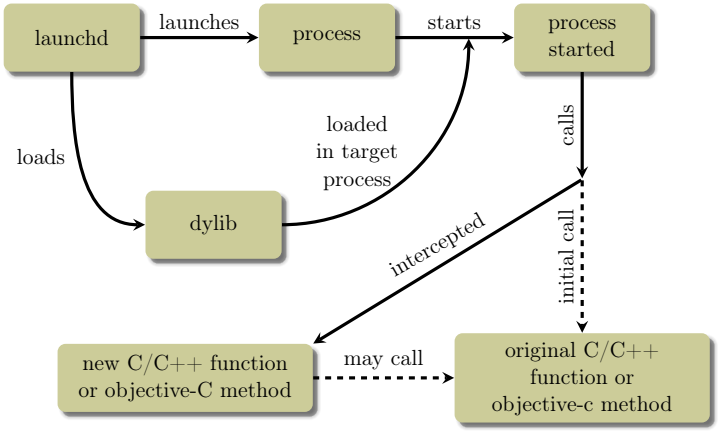


Figure 2.3: Loading *dylib* into a process on iOS.

Java frameworks are compiled before the Android application frameworks. As AIDL is part of the Android application framework, the code added (by MobileAppScrutinator) in modified core Java frameworks cannot use AIDL to send data to MobileAppScrutinator system application. Therefore, to send data to MobileAppScrutinator system apps from modified APIs of core Java frameworks, MobileAppScrutinator uses socket APIs of libc library. to send data to a dedicated service running inside MobileAppScrutinator system application. Fig. 2.1 provides a broad picture of how MobileAppScrutinator is implemented on Android OS.

2.3.2 Implementation on iOS

Fig. 2.2 depicts an overall picture of implementation of MobileAppScrutinator on iOS. The APIs of interest in iOS frameworks are modified to capture and send data related to application context as well as the PII being accessed, modified, or transmitted (clear-text or SSL). The data communication between various processes running our custom code and the daemon is through *mach* messages. In order to execute self-signed code and get privileged access to the system, the default iOS software stack needs to be modified to remove the restrictions imposed by Apple (a technique known as "Jailbreaking" in the iOS world).

On iOS, developers may write code in C, C++ and Objective-C languages. In fact, all iOS executables are linked with the Objective-C runtime [18] and this runtime environment provides a method called *method_ setImplementation*. We use this method to change the implementation of existing Objective-C methods whereas to change the implementation of

C/C++ functions, we use the trampoline technique [30]. MobileSubstrate [17], an open-source framework, greatly simplifies this task. Finally, the source code responsible for modification of various APIs of interest is compiled into a dynamic library (*dylib*) which is loaded using *launchd* [14], into all or a subset of running processes. Fig. 2.3 depicts how a *dylib* is loaded into a process using *launchd*.

2.3.3 Post-analysis of SQLite Data

The events stored in local SQLite database are processed by automated Python scripts. It is a two-step process: a first pass over the database on a per-application basis results into a JSON file, and a second pass over the JSON file derives various statistics.

Our first level analysis consists of the following steps:

1. Find all types of PII accessed by each application.
2. Check if PII is really sent over the network or not, and if yes, to which server it is sent to.
3. Search for the PII in the input to data modification APIs (cryptographic and hashing) and if found, look for the result in the data sent over the network.

First-level analysis results into a JSON file that stores 1) accessed PII, 2) PII passed to encryption or hash APIs 3) (un)modified PII sent over the Internet in cleartext or using SSL. Once the first pass over the database is finished, the resulted JSON file containing per-App details is processed to infer or derive various statistics. Here it is worth to mention that various PII accessed by an application are searched only in the network traffic and hash/cryptographic calls of that specific application.

In various APIs, the access to data is at byte level and in this case, the raw bytes are first attempted to be decoded using UTF-8 encoding. Since a different encoding may be used or in case of binary data, the hexadecimal representation of these raw bytes is also stored alongside. Searching in the network, or in the input to cryptographic or hash APIs is done for both UTF-8 encoded data and hexadecimal representation of the raw bytes.

2.3.4 Limitations

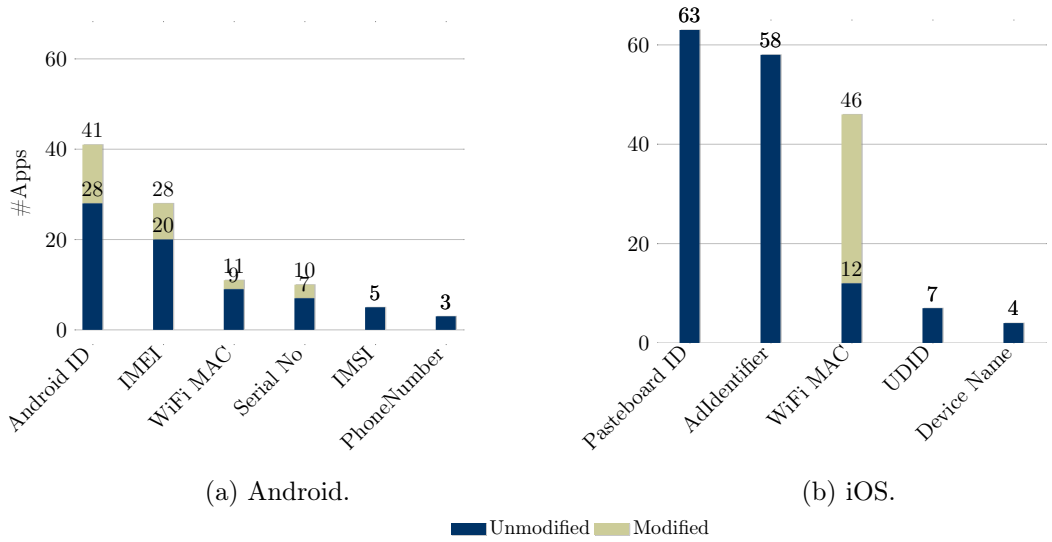
The PII leakage would remain undetected if the data is modified by the application developer using custom data modification functions before sending it over the network. If the PII is modified using OS provided data modification APIs (e.g., encryption, hashing) before sending over network, MobileAppScrutinator would correctly be able to detect the PII leakage. As an App developer is not bound to use system provided hash and encryption APIs, MobileAppScrutinator might miss some PII leakage instances.

In addition to this, the current AppScrutinator implementation on Android only supports the Java code. Therefore it currently does not detect PII leakage that involves calls to C/C++ APIs using the JNI framework. However, this is not a limitation of the approach and MobileAppScrutinator could easily be extended to handle such cases. handle JNI calls.

2.4 Experimental Setup

In order to investigate the tracking mechanisms being used by third-parties, we test 140 representative free apps available both on Android and iOS (most popular Apps in each category). Experiments have been conducted on iOS 6.1.2 and Android 4.1.1_r6.

We manually ran applications for approximately one minute each. We could interact with some applications during this one minute duration as others required the user to log



(a) Android. (b) iOS.

Figure 2.4: # Apps sending System identifiers (out of 140 apps).

in or sign up. We did not sign up or log in as our ultimate goal was not to track the manually entered user PII but the seamless background tracking done without any user intervention/interaction. Also, we did not try to cover all possible execution paths, indeed third-party library code generally starts execution when the application is first launched.

Apart from device or operating system unique identifiers and information, we also entered other synthetic information such as addressbook, calendar events, accounts etc. This enables us to know if such data is accessed and transmitted by apps.

2.5 Cross-App Third-Party Tracking

Smartphone users mostly use dedicated apps rather than websites for accessing services, essentially because of the relatively small screen size and the lack of mobile-optimized web pages (even if this later aspect has largely improved). Therefore user tracking is no longer performed only through "third-party cookies" in web browsers but also in apps through dedicated identifiers. In this section, we shed some light on this ongoing user tracking in apps through stable identifiers.

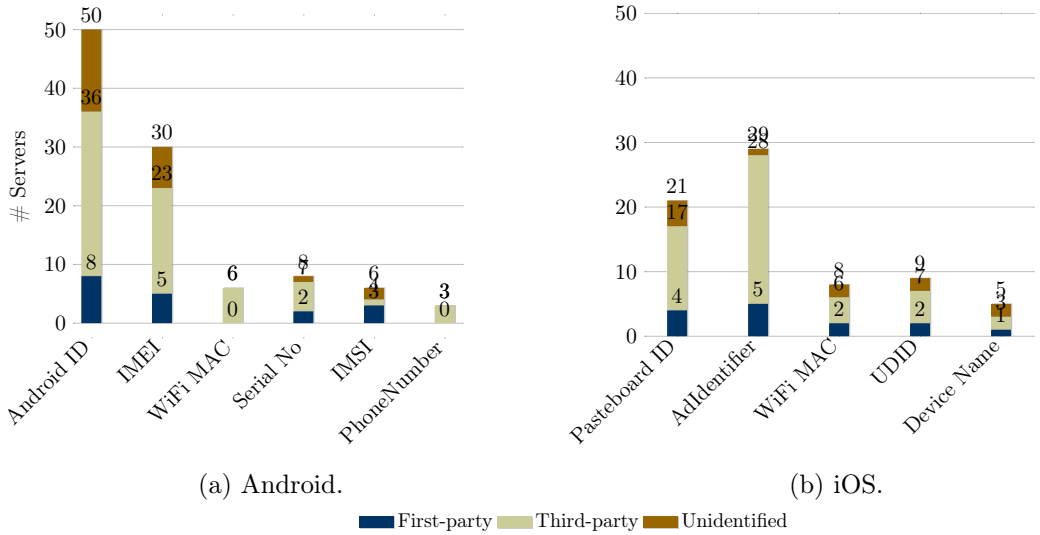
2.5.1 Unique identifiers from the system

First of all, let us consider the system level unique identifiers. The situation is rather different depending on the target OS.

Android.

Various system identifiers are available on Android. A user permission is required to access hardware-tied identifiers (IMEI and Wi-Fi MAC address) as well as SIM-tied identifiers (IMSI and phone number). OS-tied identifiers (Serial Number and AndroidID) identifiers are freely available to be accessed.

Fig. 2.4a presents the number of android apps (out of a total of 140 most popular apps tested) that transmit various unique system identifiers over the Internet. We note that a significant number of apps transmit hardware-tied identifiers such as IMEI number and WiFi MAC address of the phone. IMEI number alone is transmitted by 28 apps, i.e., by 20 % apps. This is very critical in terms of user privacy because users cannot reset hardware-tied identifiers (unlike a cookie in a web browser). Android ID is the most frequently



(a) Android. (b) iOS.

Figure 2.5: # servers where system identifiers are sent by 140 apps.

OS-tied identifier transmitted (by around 30% apps). In general, SIM-tied identifiers are the least frequently transmitted: IMSI is transmitted by five apps whereas three apps transmitted the phone number. Interestingly, we also note that most of these identifers are transmitted unmodified, i.e., without hashing or encrypting, over the Internet. This clearly demonstrates that app developers do not care at all about user privacy and the legislation is not hard enough to force the app developers to care about user privacy.

Fig. 2.5a presents the number of servers where unique system identifiers are transmitted by 140 Android apps. The servers are classified as either first-party or third-party or unidentified. First-party servers are those where domain name conincides with the app name, third-party servers are domains of well known advertisers and trackers, and unidentified are the ones where domain name is either of content providers or ip addresses for which no hostname information is found. We note here that various unique system identifiers are transmitted to both first and third-parties. We also find that various unique identifiers are also sent to some IP addresses without any hostname information. It is not easy to identify to whom such machines belong to and why these identifiers are transmitted to them. However, we find that third-parties collect these unique identifiers more often than first-parties. In fact, depending on the app permissions, third parties try to collect as many identifiers as they can: for instance, third-party domains like *ad-x.co.uk*, *adxtracking.com* and *mobilecore.com* all collect and send the AndroidID, IMEI and WiFi-MAC address to their servers in clear-text. Additionally, first and third-parties both send frequently these unique identifiers over the Internet to their servers unmodified (e.g., without hashing) and in clear-text (without SSL). This is a serious threat to user privacy as a network eavesdropper can easily correlate the data flowing through the Internet. For interested raders, Table A.1 in the appendix A of this thesis provides the whole list of servers and the corresponding unique identifiers transmitted to them by 140 Android apps in clear-text or through SSL.

Along with these unique system identifiers, third-parties also collect and send the names of apps in which their code is present. We notice that User-Agent http header field generally contains package/App name. As knowing the apps of a user reveals a lot about user interests [188] and increases the re-identification risk [60], this is a serious privacy threat. In fact, the collection of such data is proportional to the number of apps in which third-party code is present and the number of apps sending these unique identifiers. So it is interesting to quantify the number of apps sending these unique identifiers to third-parties. Looking

at both Fig. 2.5a and Fig. 2.4a, it can easily be deduced that the presence of third-parties in Android Apps is huge. Globally, we find that 31% (44 out of 140) of Apps transmit, at least, one (un)modified unique identifier over the Internet.

iOS.

Fig. 2.4b presents the number of iOS apps (out of a total of 140 most popular apps tested) that transmit various unique system identifiers over the Internet. The "AdIdentifier" is transmitted the most, which is fine because it has been specifically introduced for Advertizing and Analytics purposes as a replacement to the deprecated UDID. However we notice that some companies (e.g., tapjoyads.com, greystripe.com, mdotm.com, admob.com and ad-inside.com) are still using deprecated UDID. We also found four apps collecting the device name. In iOS, the device name (DeviceName) is set by the user during the initial device setup and often contains the user's real name. Since this device name is stable (the user generally does not modify it), even if it is not guaranteed to be unique across all devices, it is a stable identifer that can probably be used for tracking purposes. Additionally, if it is set with the user's real name, it may reveal user identity. We also notice that these identifiers are always collected when the user starts/stops interacting with the app. This means that third-parties can even know how long a user is using a particular app and the time when a user goes idle, revealing user habits. As, globally, 60% (i.e., 84 out of 140) of apps send, at least, one (un)modified unique identifier over the Internet, the risk in terms of privacy is huge.

Fig. 2.5b presents the number of servers where unique system identifiers are transmitted by 140 iOS apps. We find a lot of apps transmitting AdIdentifier over the Internet to their servers as well as third-parties. UDID is still being used and transmitted by a total of 9 apps. Surprisingly, the device name is transmitted to 5 servers. Out of these 5 servers, only 1 server belongs to a first-party whereas others are either third-party or identified. In case of iOS, Wi-Fi MAC address is the only hardware-tied identifier that is available to be accessed. It is transmitted to both first and third-parties. More details about servers where these unique identifiers are sent, can be found in Table A.2 in the appendix A of this thesis.

2.5.2 Unique identifiers generated by third-parties

Let us now consider unique identifiers generated by third-parties in order to bypass OS restrictions. As access to system unique identifier is limited on iOS, third-parties generate and share unique identifiers to have a mechanism to track users across apps. However, apps are sandboxed on iOS (as well as on Android) and therefore, these third-parties cannot simply share those generated unique identifiers across apps. To circumvent this limitation on iOS, third-parties use a class called UIPasteBoard [32]. This is specifically designed for cut/copy/paste operations with the ability to share data between apps. The data shared by apps with this class can be persistently stored even across device restarts. Among the Apps we tested, we found that a large number of third-parties use the UIPasteboard class to share a unique third-party identifier across apps. Looking at the names and types of pasteboards created and the servers where these values are sent, we found that 63 Apps create at least one new pasteboard entry at the initiative of a third-party library (Fig. 2.4b).

Essentially, third-party code present inside an application stores a pasteboard entry with its unique name, type and value. Later, if an App containing the code from the same third-party is installed, it retrieves the value corresponding to its pastebaord name. To have a look on pasteboard names, types and values used by various third-parties present inside 140 iOS Apps tested, please refer to Table A.7 in the appendix A of this thesis. Here it is to be noted that user has no control over this kind of tracking and "Limit Ad

Tracking" feature of iOS is ineffective in this case.

From Fig. 2.5b, we note that these pastboard entries are transmitted to various first and third-parties even though pastboard entries are just designed for cut/copy/paste operation between apps. Moreover, we find that pastboard entries are more transmitted to third-parties than first-parties. This assures our assumption that third-parties generate these unique pastbaord entries to track users because they transmit them over the Internet.

2.5.3 Comparation of third-party tracking on Android and iOS

Comparing the identifiers transmitted by 140 most popular apps on Android and iOS reveals that iOS apps transmit system identifiers more often than Android apps. However, iOS apps mostly transmit dedicated identifier, i.e., advertising id, for tracking and advertising purposes and not many apps transmit hardware-tied system identifiers as compared to app on Android. This is probably due to the fact that many system identifiers (hardware and SIM-tied identifiers like IMEI, IMSI, and Serial number) are not available to be accessed on iOS.

As opposed to iOS apps, we did not notice Android apps generating and sharing unique systems identifiers for tracking purposes. This is probably because various system identifiers are readily avaialble to be accessed by apps on Android and therefore, third-parites do not need to generate their own identifiers. Indeed, identifiers such as serial number and Android ID do not even need a permission to be accessed. As tracking through system identifiers is more reliable and accurate, trackers probably do not need to resort to this solution.

In conclusion, Android makes available wider range of system unique identifiers to apps as compared to iOS therefore it is easier for third-parties to track Android users as compared to iOS. Nevertheless, this does not stop trackers on iOS which resort to other techniques.

2.6 Collection of other personal information

To create a rich user profile, third-parties can use various means to collect a wide variety of personal information:

1. By directly collecting as much information as possible from the device (i.e., by adding the appropriate code in the libraries to be included by the app developers).

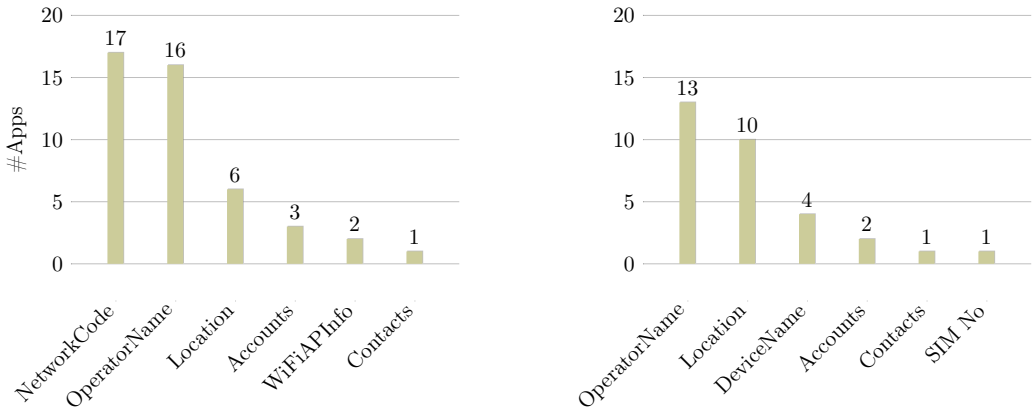
2. By retrieving it from other third-parties who have already collected this information (thereby, aggregating the user PII [24]).

3. By obtaining it from first-parties.

It is difficult to measure how much information are being shared among third-parties themselves or among first and third-parties, but we next measure in this section what kinds of data and to which extent are being collected by these third-parties directly from the smartphone.

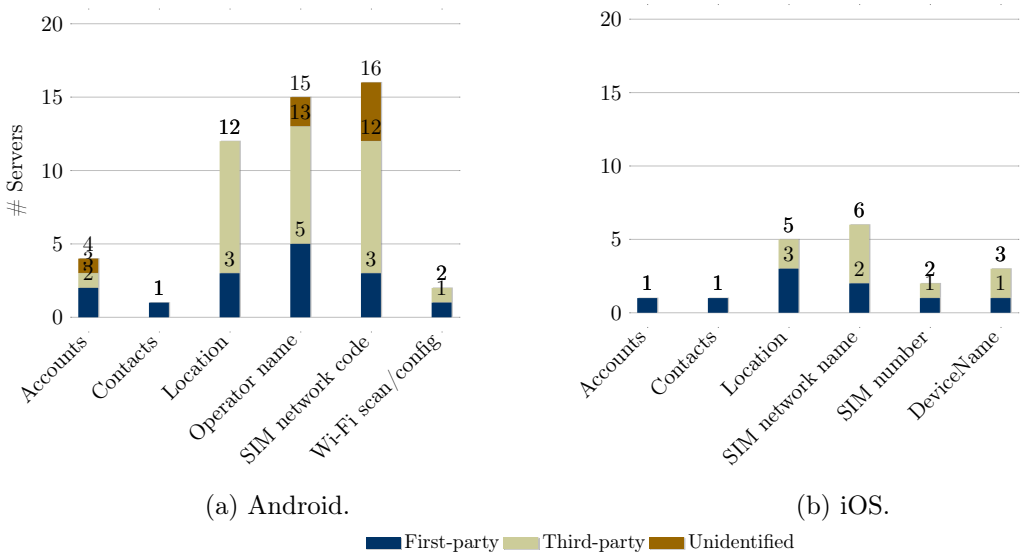
2.6.1 Android

Various personal data of the user is available to be accessed for apps on Android so that various useful apps can be developed. Such personal data includes, e.g., location of the user, contacts or the accounts. Our experiments with 140 Android apps reveal that different kind of user personal information is being sent over the Internet to various first and third-parties.



(a) Android. (b) iOS.

Figure 2.6: # Apps sending PII out of a total of 140 apps.



(a) Android. (b) iOS.

Figure 2.7: # servers where user personal information is sent by 140 apps.

Fig. 2.6a presents number of apps sending different kind of data over the Internet. We see that network code and operator name is sent by 17 and 16 apps respectively. User location and accounts information is transmitted by six and three apps respectively. Two apps transmit information related to the Wi-Fi access point they are connected to whereas one app transmit user contacts over the Internet.

Fig. 2.7a reveal that user location is transmitted (encrypted or in clear-text) to nine third-parties whereas to three first parties. In fact, it is more often sent to third-parties than first-parties. We also find that user location is sent (encrypted or in clear-text) to nine third-parties whereas it is sent to only three first-parties. This means that user location is used more often for tracking and profiling the user and not for providing a useful service. Otherwise, we also note that the name of the telephony operator and the SIM network code is being collected by a lot of first and third-parties (details available in Table A.5 in the appendix A of this thesis).

Moreover, as the apps installed on a device is highly valuable information for trackers/advertisers to infer user interests and habits, we detect and measure the leakage of this information too. We find that 5 third-parties know 4 or more apps installed by a user. This puts users at serious privacy risks as users can be re-identified later with high

probablity [60]. Specifically, "tardemob.com", present in "Booking.com" App, collects the list of all apps installed on the device and sends this list to its server (details available in Table A.8 in the appendix A of this thesis).

2.6.2 iOS

iOS also makes accessible many user PII sources (e.g., Accounts, Location, or Contacts) to apps. This is necessary so that a wide variety of apps can be developed.

Figure 2.6b presents the number of apps sending user PII over the Internet. We find that 10 apps (out of 140) send user location over the Internet. SIM network (operator) name and SIM number are transmitted most by iOS apps. Device name is transmitted by four apps.

Fig. 2.7b shows whether the location data it is sent to first or third-parties by these 10 apps. We find that it is sent to two third-parties (to one in clear-text and one using SSL) and three first-parties. SIM network name is mostly transmitted to third-parties (details about the transmitted user data and where it is sent to are available in Table A.6 in the appendix A of this thesis).

2.6.3 Comparison between Android and iOS

We find that 10 apps (out of 140) send user location over the Internet as compared to 6 apps on Android. However, more third-parties collect and send user location over the Internet on Android. This means that on iOS, there are less third-parties but they are more broadly used by apps, on the other hand there are more third-parties in Android used by less apps.

As opposed to Android, we find that iOS apps leak less user PII. In total, there are 8 (as opposed to 21 on Android) third-parties where user PII is sent to on iOS. Also, both first and third-parties did not send much data to their servers in clear-text. There is only one third-party server and one first-party server where user location is sent in clear-text on iOS as compared to six and one respectively on Android. Globally, we note that iOS apps sent lesser user PII over the Internet as they use SSL more often than their Android counterparts.

iOS apps also leak more information about the list of installed apps on the phone as compared to Android apps. Nine third-parties know, at least, five names of the installed packages. Flurry, for example, knows 25 apps installed on the phone and is included in all these apps. The included library from Flurry sends the name of the app in which it is present as part of the communication with their servers. Moreover, the collection of this information is in plain-text. To get the complete list of these third-parties as well as the package names known to them, please refer to Table A.9 in the appendix A of the thesis.

2.7 Discussion

2.7.1 Comparing MobileAppScrutinator on Android with TaintDroid

Fig. 2.8 compares privacy leaks reported by MobileAppScrutinator on Android and TaintDroid for some personal data. To know more details, please refer to the Table A.3 in the appendix A of this thesis that presents the PII (unique identifiers and other private information) leaks reported by running the same set of applications on TaintDroid 4.3. We note that TaintDroid only reported the leakage of one unique system identifier (IMEI) whereas MobileAppScrutinator reports the leakage of six different unique identifiers. Moreover, MobileAppScrutinator overall reported more privacy leaks than TaintDroid. However, we interestingly found that both MobileAppScrutinator and TaintDroid have false negatives.

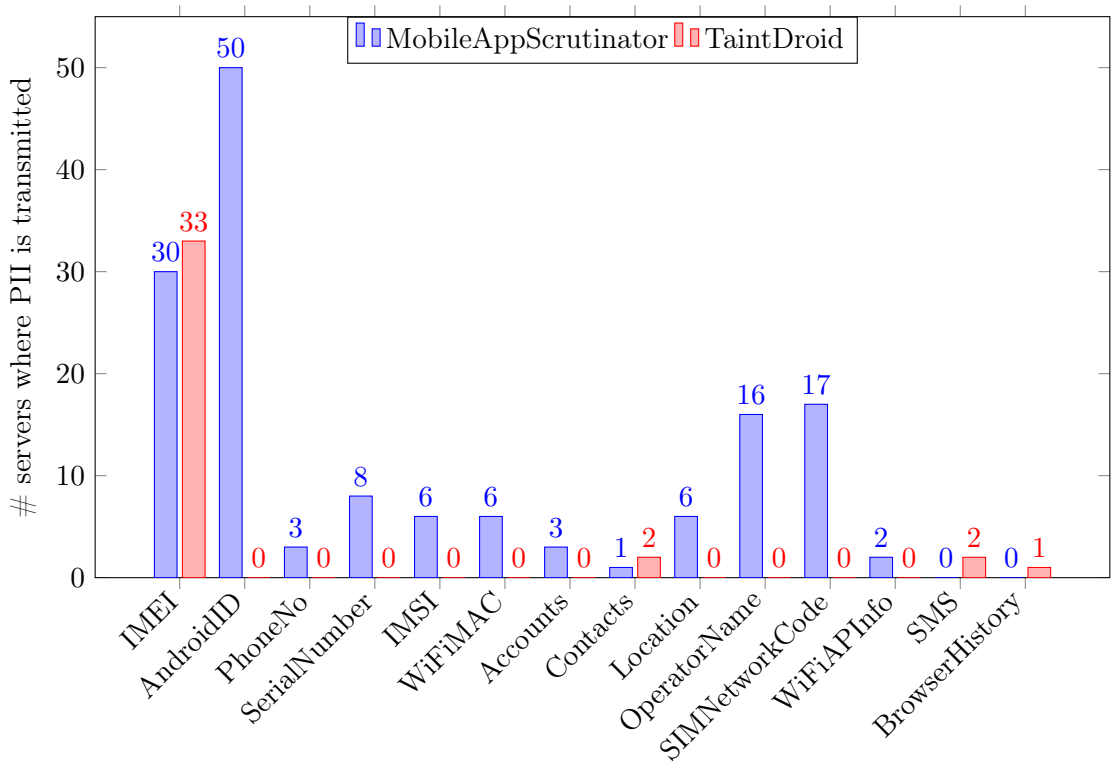


Figure 2.8: Comparation of privacy leaks reported by MobileAppScrutinator on Android and TaintDroid

This suggests that these tools should not be replacement of one another but can actually be complimentary to each other.

While comparing PII leaks reported by MobileAppScrutinator with TaintDroid, we found that TaintDroid did not report any leakage of location coordinates. TaintDroid reported leakage of Address Book (Contacts) information to two third-parties whereas MobileAppScrutinator could only detect Contacts leakage to only one party. Again it is interesting to note here that parties, where Contacts information was leaked, are mutually exclusive. TaintDroid also reported the leakage of Browser History and SMS to one and two parties respectively but, as MobileAppScrutinator did not implement their leakage detection, we cannot compare with respect to these two types of PII. On the contrary, MobileAppScrutinator was implemented to detect the leakage of other kinds of private data, such as Accounts, Operator Name, SIM Network Code and WiFi Scan/Config info, which current implementation of TaintDroid (version 4.3) lacked.

2.7.2 Effectiveness of various privacy safeguards

In order to provide transparency and control over privacy, both Android and iOS involve user decisions along with mechanisms adopted by their respective systems. However, the approach followed by Android and iOS is different: Android employs a static install-time permission system whereas iOS solicits explicit user permission at runtime. No doubt these OS mechanisms are mostly effective, they lack behavioural analysis, i.e., when, where and how often the accessed information is sent over network. For example, it is vital to distinguish the fact if the PII is sent to an application server or to a remote third-party. In fact, a user giving access to her PII for a desired service does not necessarily mean that she also wants to share this information with other parties, for example, advertisers or analytics companies. Similarly, an application accessing and sending user location only at

installation time is not the same as sending it every five minutes.

Below we discuss the effectiveness of various privacy safeguards available on both Android and iOS based on our experiments and results.

Resetting the "AdIdentifier" on iOS The effect of resetting the AdIdentifier is not similar to "Deleting the cookies" in web tracking and could easily be nullified. Resetting the AdIdentifier, in theory, is meant to prevent trackers from linking the user activity before and after the reset. However the trackers can easily detect the AdIdentifier change and link the two values even if Apple explicitly tells not to do so. Apps are not technically restricted by iOS to respect the resetting of AdIdentifier on iOS. In our study, we find that 20% apps send the IdentifierForVendor[2] along with the AdIdentifier to third-parties (Details in Table A.4 in the appendix A of this thesis). It is noticeable that many third-parties collect this identifier, whereas it was principally designed by Apple to be used only by first-parties. As IdentifierForVendor is being collected by third-parties, they are able to link the AdIdentifiers before and after the reset.

Apps bypass the "AdIdentifier" on iOS We have seen that many apps are using other tracking mechanisms to track the user in addition to the AdIdentifier. In our experiments, we discovered that 93 apps out of 140 (i.e., approx. 66%) will continue to track the user after a reset of the AdIdentifier by the user. This measurement does not even consider the applications employing the previously described technique to match the changed/reset AdIdentifiers as we cannot be sure what third-parties do with their data collected. In iOS 7, Apple banned the access to WiFi MAC Address, but the percentage only reduces from 66% to approximately 42% (60 Apps out of a total of 140 Apps.), i.e., if we exclude the apps (24%) using only WiFi MAC address as a unique identifier for tracking.

"Limit Ad Tracking" iOS privacy setting It turns out that the "AdIdentifier" is available to all Apps, even after a user has chosen the "Limit Ad Tracking" option. It is therefore ambiguous how iOS can enforce the "Limit Ad Tracking" option. In fact, "Limit Ad Tracking" setting appear to be misleading, as it gives the end-user a wrong feeling of opting-out from device tracking. And if recently Apple started to reject apps accessing the AdIdentifier without providing In-App Advertising [3], which is reasonable, it still does not solve the core problem.

2.8 Conclusion

This chapter introduces the MobileAppScrutinator platform for the study of third-party smartphone tracking. To the best of our knowledge, this platform is the first one that embraces both iOS and Android, using the same dynamic analysis methodology in both cases. For the first time, it provides in-depth insight on the different PII accessed, hashed and/or encrypted and sent to remote servers, either in clear-text or over SSL connections. This in-depth analysis capability is a key to analyse the applications and understand how they (ab)use personal information.

The second major contribution of this work is the behavioural analysis, thanks to the MobileAppScrutinator platform, of 140 free and popular apps, selected so that they are available on both mobile OSes in order to enable comparisons. Two important aspects are considered: first we show that many stable identifiers are collected on Android, in order to track individual devices in the long term. On iOS, availability of system-level identifiers is

[2]This is a stable identifier unique to all apps of a single developer on a particular device, that cannot be changed or reset by the user.

less common, but techniques have been designed to create new cross-app, stable identifiers by third-parties themselves. The second aspect concerns the user-related information. We show that a significant amount of PII is being collected by third-parties who implicitly know a lot about the user interests (e.g., by collecting the list of apps installed or currently running).

Finally, this work enables to have a comparative view of ongoing tracking on Android and iOS. Our experiments show that Android apps are more privacy-invasive when compared to iOS apps as the presence of third-parties is clearly more frequent in Android applications. In all cases, protective measures should be taken by device manufacturers, OS designers and various regulatory authorities in a coordinated way to control the collection and usage of PII.

Chapter 3

User Tracking through Innocuous Android Permissions

Contents

3.1 Introduction

Today a large fraction of mobile devices are running Android, where access to user data is controlled by the permission system. The goal of this permission system is to let users decide if they want to install an application or not based on the list of required permissions. It is worth mentioning that an application can only be installed if the user agrees with the required list of permissions. However, users have a poor understanding of the permissions and do not pay enough attention to these messages; they therefore often do not realize the kind of information accessible to an application once installed [109]. In addition, the permission system sometimes fails to prevent the access to sensitive data [208].

In order to help users perceive the potential risks implied by the permissions, Android classifies the permissions based on their expected risk[1]. `ACCESS_WIFI_STATE`, on which we focus in this work, allows an application to access various information related to the the Wi-Fi interface. As such it is categorized as `normal` as compared to, e.g., `ACCESS_FINE_LOCATION` which is classified as `dangerous`. In this work we show that the `ACCESS_WIFI_STATE` permission is actually `dangerous` as a user's PII can be (and is already) derived from the use of this permission. Also, as the use of this permission by ad libraries has increased over time [79], the problem is severe.

[1] `http://developer.android.com/guide/topics/manifest/permission-element.html`

Table 3.1: `WifiManager`'s method restricted by the `ACCESS_WIFI_STATE` permission.

Method name	Description	Retrievable Information
isWifiEnabled()	Returns whether Wi-Fi is enabled or disabled.	Returns true if Wi-Fi is enabled.
getWifiState()	Gets the Wi-Fi enabled state.	(Currently being) enabled/disabled or unknown
getConfiguredNetworks()	Returns a list of all configured networks.	**For each configured network/AP**: SSID, allowed protocols and security schemes
getConnectionInfo()	Returns dynamic information about the current Wi-Fi connection, if any is active.	**About AP:** BSSID, SSID, RSSI **About Device:** Wi-Fi MAC address, IP address
getScanResults()	Returns the results of the last AP scan.	**For each AP:** BSSID, SSID, signal strength, channel, capabilities
getDhcpInfo()	Returns the DHCP-assigned addresses from the last successful DHCP request, if any.	IP address, DNS server address, gateway and netmask

Even though, we focus on `ACCESS_WIFI_STATE` permission in this work, there are also other permissions that have hidden privacy implications. For example, `READ_PHONE_STATE` permission gives access to a hardware-tied identifier, IMEI for GSM and the MEID or ESN for CDMA phones. There has been a lot of discussion about separeting phone state reading and unique identifiers on issue tracker website of Android Open Source Project (AOSP) but the issue is not resolved [25, 9]. Other unique identifiers on the Android are also available to be accessed either without requiring a permission or using a permission that does not explicitely state the availability of unique identifiers [20].

More specifically the contributions of this work are threefold: First, we consolidate what PII can be derived from the data related to Wi-Fi interface (Section 3.3), namely unique identifiers (useful for tracking purposes), device geolocation, travel history and social links between users. Second, we analyze the current situation on Google Play employing both static and dynamic analysis of Android applications (Section 3.4), revealing that a large number of applications have actually started to exploit this permission to obtain user PII. Finally, we analyze the user perception of this permission using an online survey to which 156 Android users answered (Section 3.5). The results clearly demonstrate that users do not understand the privacy implications of this permission.

3.2 Background and Related Work

3.2.1 Android Permission System

As Android gives no privilege to applications by default, applications must ask the user for privileges by statically declaring the list of permissions it requires. At installation time, the Android system prompts the user for consent, and the installation completes only if the user agrees with the permission an app requires.

There are a total of 145 different permissions available (as of Android version 4.4) for an application to ask for. Many of these permissions are required by applications to access user sensitive information (e.g. `ACCESS_FINE_LOCATION` and `READ_CONTACTS` are the permissions required to respectively geolocalize the device and read user's contact data). The permissions are also categorized, based on the associated potential risks, as either `normal` or `dangerous`.

The ACCESS_WIFI_STATE permission. It makes some of the methods of the WifiManager class[2] available to be accessed by applications (Table 3.1) and falls in the 'Network communications' group of permissions. It is worth mentioning that these methods only allow to read the data associated with the Wi-Fi interface, but not to modify it: a different permission, CHANGE_WIFI_STATE, is required to change the state of the Wi-Fi interface. ACCESS_WIFI_STATE permission is categorized as normal (we will see in later sections that this decision is highly questionable). Also, it is important to realize that this is not the permission required for Internet access. INTERNET is the only permission required to open a network socket to send or receive data over the network.

3.2.2 Related Work

Inference of private information by application on the Android system have been studied in [208]. This work focuses on the exploitation of *public* information available on the the Android system, i.e., data that do not require any permission to be accessed. In particular, they show that some PII (such as, geolocation, driving route, identity) can be inferred from the *public* information made available to be accessed for applications by the Android system. We note that getting device geolocation is common in both ours and [208]. However, in [208], device geolocation is only obtained when the device is connected to a Wi-Fi network, whereas in our study, we show that it can be obtained as long as the Wi-Fi interface is enabled.

The problem of overprivileged applications have been studied in [108]. According to [108], a third of the applications are requesting more permissions than they do actually require. One of the proposed explanation is the confusion between permission name, which is typically the case for network related permission such as ACCESS_WIFI_STATE and ACCESS_NETWORK_STATE. Our results are in line with those findings as we have identified 17% of the applications requesting ACCESS_WIFI_STATE permission without really using it.

Book et al. [79] investigates changes over time in the behavior of Android ad libraries. The study reveals that number of libraries able to use different permissions, ACCESS_WIFI_STATE permission is one among them, drastically increased over time. Also, Nguyen et al. [139] show that Wi-Fi information could be used to breach location privacy. However, [79] and [139] did not analyze the current situation on Google Play. As opposed to these studies, our study employs in-depth analysis of applications requesting ACCESS_WIFI_STATE permission and shows that a number of user PII can be and are already derived from the data accessible using this seemingly network-related permission.

The user comprehension of Android permissions have been studied in [109]. Similar to our work, this study includes a questionnaire to evaluate the understanding of the permissions. The study considered a total of 11 permissions including READ_PHONE_STATE and CHANGE_NETWORK_STATE, but left out ACCESS_WIFI_STATE and the ACCESS_FINE_LOCATION permissions. Likewise our specific study of ACCESS_WIFI_STATE permission, the results of this study also indicate that users have a poor understanding of the Android permission system.

3.3 User PII Inferred from Wi-Fi Data

As we have seen, the ACCESS_WIFI_STATE permission enables an application to read data related to the Wi-Fi configuration of the device. This raw data may look innocuous, but it is actually possible to either directly access or infer several user PII. In this section, we describe such user PII.

[2] http://developer.android.com/reference/android/net/wifi/WifiManager.html

A unique device identifier. Using the `getConnectionInfo()` method, an application can obtain the MAC address of the Wi-Fi interface. In fact, there are other hardware-tied identifiers available to be accessed by applications, e.g., `IMEI` and `MEID` with `READ_PHONE_STATE` permission. As these unique identifiers could be used by advertisers to track user activities across all applications, they pose serious threat to user privacy. In particular, Wi-Fi MAC is also used to track users in the physical world [161, 95] and therefore, allows trackers to link both online and physical profiles of the user.

The Wi-Fi MAC address is used by tracking systems that monitor each wireless channel to track individuals as they move in the physical world (e.g., in a retail store of a shopping mall) [161, 95]. If in parallel it is also collected by applications, then it paves the way for trackers to create a bigger profile based on both the online and physical activities of the user. Indeed, this technique can be used to detect the user and to serve targeted advertisement in the physical world based on the online profile of the user [29]. It is quite evident from the above discussion that the availability of Wi-Fi MAC address to advertisers is a serious threat to the user privacy.

Geolocation. Geolocation is traditionally obtained from the dedicated GPS chip embedded in the device. In addition to this, Wi-Fi and GSM based geolocation systems are also used to improve accuracy and decrease the first fix time of the geolocation or as an alternative to the GPS to reduce battery life. These systems use the information about surrounding Wi-Fi APs (resp. GSM cell towers) to derive device geolocation.

The list of surrounding Wi-Fi APs can be obtained thanks to the `ACCESS_WIFI_STATE` permission through the `getScanResults()` method of `WifiManager` class. This method does not trigger the scanning of Wi-Fi APs, but return the list of last scanned Wi-Fi APs. Wi-Fi scan is performed automatically by the system every 15 seconds and can also be triggered by other applications, therefore this list of surrounding Wi-Fi APs is often up-to-date.

By submitting the raw result of a Wi-Fi scan to a remote geolocation service[3], the device gets geolocation information (coordinates and accuracy metric) in return. Wi-Fi based geolocation is accurate to 20 meters in urban areas [138]. This is due to both the high density of Wi-Fi APs in urban areas and a coverage area comparatively smaller than GSM Cell towers. The Wi-Fi, GSM and GPS-based geolocation systems are employed by the Android system, but access to this information is restricted by the `ACCESS_FINE/COARSE_LOCATION` permissions. On the opposite, the raw Wi-Fi scan information is not protected by any of these two geolocation permissions. By using a third party Wi-Fi based geolocation service, it is therefore possible for an application to obtain geolocation information without having to explicitly ask for geolocation permissions as long as the application has both the `ACCESS_WIFI_STATE` and `INTERNET` permissions. And in Section 3.4, we show that this is often the case.

Travel history. The list of Wi-Fi networks to which the device has been connected to in the past is stored in the *Configured Networks List* that can be accessed through the `getConfiguredNetworks()` method of `WifiManager` class. For each of these configured networks, the SSID is available along with other information such as the supported security protocols and authentication algorithms used. It has been shown in [120] that the data stored in the *Configured Networks List* can be combined with external resources[4] to obtain information such as the previously visited locations.

[3] `https://developers.google.com/maps/documentation/business/geolocation/`
[4] `http://wigle.net/`

Table 3.2: Most commonly used methods of `WifiManager` class, in 998 applications.

Method call	# of Apps
getConnectionInfo()	753 (75.45%)
isWiFiEnabled()	344 (33.47%)
getScanResults()	156 (15.63%)
getConfiguredNetworks()	59 (5.91%)
getWifiState()	76 (7.62%)
getDhcpInfo()	63 (6.31%)

Social links. It is possible to predict the existence of a social or professional link between the owners of devices by comparing the list of SSIDs known by any two devices [93]. By collecting this data on a large population, an application could gather information that would make it possible for them to build a social network between their users. The *Configured Networks List* returned by the `getConfiguredNetworks()` method can be used for the same purpose. In fact, it also contains some other information about the configured Wi-Fi AP (e.g., allowed protocols, authentication algorithms, key management). This information can be leveraged to improve the quality of the social link establishment as [93] relies only on the SSID.

Other PII derived from SSIDs. SSIDs are often made to be meaningful to users and potentially contain information about the network owner or its users (e.g., name of institutions, individuals or locations [146]). Extraction of this information can be done manually or could be automated using techniques like Named Entity Recognition [92] and could then be used, for example, by advertisers to enrich the profile of the user to serve targeted ads.

3.4 Android Applications Analysis

We have analysed the 100 most popular free applications in each of the 27 categories present on Google Play, i.e., 2700 applications in total. We focused only on those that require both the `ACCESS_WIFI_STATE` and the `INTERNET` permissions (only 5 applications ask for the first permission but not the second one). To that purpose, we crawled Google Play[5] and collected the list of required permissions. Then we statically analysed 998 applications requesting these permissions and based on the results, we chose 88 applications for an in depth, dynamic analysis. This section details our findings.

3.4.1 Static analysis

We used custom scripts (based on Androguard[6]) to statically analyze the 998 APK files corresponding to applications requiring both the `ACCESS_WIFI_STATE` and the `INTERNET` permissions.

Use of the `WifiManager` class methods

We note that 17% (165) of the applications request `ACCESS_WIFI_STATE` permission but do not access any of its methods. Those over privileged applications present a privacy risk as later revisions of those applications will be able to use the protected methods by this permission.

[5] Using `https://github.com/egirault/googleplay-api`
[6] `https://code.google.com/p/androguard/`

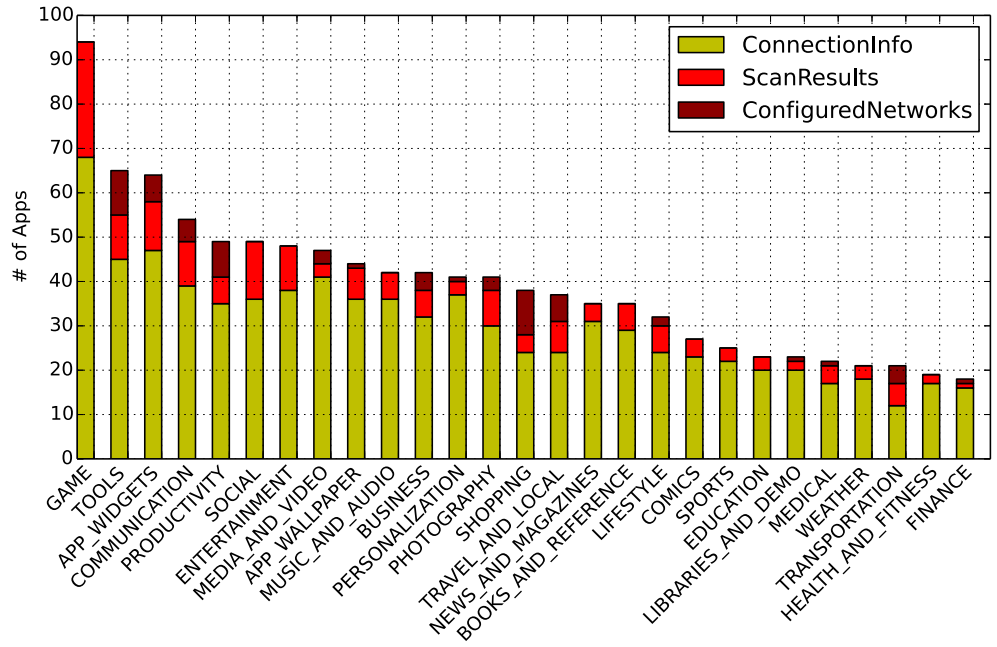


Figure 3.1: Per category popularity of 3 privacy-sensitive methods (100 applications/category).

Table 3.2 presents the number of applications calling the `WifiManager` class' methods. In fact, ∼76% (762) applications are accessing at least one of these three privacy-sensitive methods whereas ∼1% (11) of them are accessing all 3 privacy sensitive methods. Among the 6 methods protected by this permission, we chose to focus on the ones that pose serious privacy risks to the user, namely: `getScanResults()`, `getConfiguredNetworks()` and `getConnectionInfo()`. Note that the access to `WifiManager` class' methods might be legitimate, for example, in case of a Wi-Fi manager application in the `Tools` or `App Widget` categories, but probably not for a cooking or wallpaper application. From now on, our analysis only focuses on these three methods.

Figure 3.1 presents a per category distribution of the applications accessing these privacy-sensitive methods. Overall, applications in the `Game` category are the ones that use these 3 privacy-sensitive methods the most. There are also several other categories of applications that show a high usage of those methods without an obvious reason, like `Lifestyle`, `Comics` and `App Wallpaper`.

Analysis of third-party code inside applications

It is interesting to identify if the method calls are made by code written by the application developer ("first-party") or by the libraries (e.g., advertisement, analytics, performance monitors, crash reporters) included by the application developer ("third-party"). For this purpose we use a heuristic: classes belonging to a package whose name is the same as the application package name is considered coming from the application developer, otherwise it is considered as third-party code. Based on this, we find that 18% (136) of the applications accessing at least one of these methods are doing so only from third-party code. This confirms that third-party code is often responsible for the usage of the `ACCESS_WIFI_STATE` permission. Access to an API call by third-party code is sometimes legitimate, but there are cases where it is clearly not (e.g., Wi-Fi scanning is performed by `inmobi.com` and `skyhookwireless.com` in 13 applications out of 156 that scan for Wi-Fi APs).

Figure 3.2 presents, for each of the three privacy-sensitive methods, whether a first-party, third-party, or both access these methods. It reveals that there are some applications

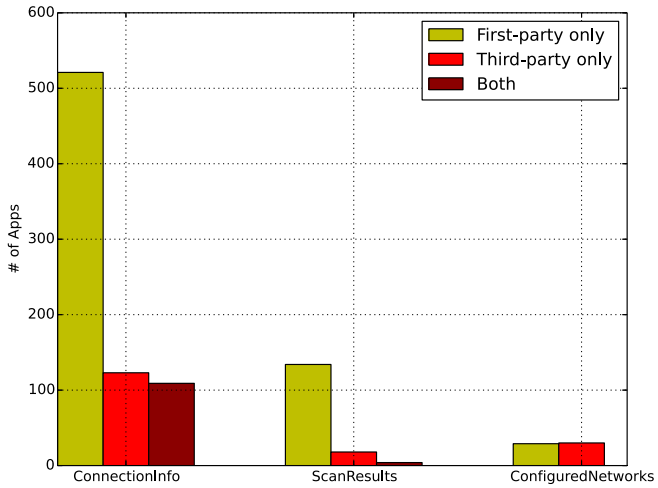


Figure 3.2: Distribution based on the party accessing privacy-sensitive methods, in 762 applications.

Table 3.3: Top 5 third-parties in each category and their corresponding number of applications.

ConnectionInfo		**ScanResults**		**ConfiguredNetworks**	
Third-party	# **Apps**	**Third-party**	# **Apps**	**Third-party**	# **Apps**
inmobi.com	74	inmobi.com	9	google.com	10
chartboost.com	55	domob.cn	9	mobiletag.com	4
tapjoy.com	49	mologiq.com	6	lechucksoft-ware.com	2
vungle.com	47	tencent.com	5	android.com	2
jirbo.com	43	skyhookwire-less.com	4	Unibail.com	1

in which only the third-party code accesses these privacy-sensitive methods. This means that if the code written by the application developer does not require the `ACCESS_WIFI_STATE` permission, the third-party library does need it. The motivation for a third party library can be to secretly collect user information, or to provide a functionality to the application. For example, an application developer or a third-party can use the code from `skyhookwireless.com` to retrieve device geolocation without needing explicit dedicated geolocation permissions. It is worth mentioning that `skyhookwireless.com` retrieves the list of surrounding Wi-Fi APs in 4 applications (Table 3.3). In any case deriving device geolocation without any explicit user permission is not legitimate and should be prevented by Android.

Table 3.3 presents the top 5 third-parties in each category and the number of applications in which they are present. Looking at the web pages of these third-parties, one may understand the purpose of these third-parties in various applications. It seems like most of them (`inmobi.com`, `jirbo.com`, `vungle.com`, `chartboost.com`) belong to A&A business, whereas others are different kinds of service providers, e.g., `skyhookwireless.com`. Here it is worth noting that `skyhookwireless.com` provides geolocation service among other kinds of services. With this service, an application can get the location of the phone with the use of `ACCESS_WIFI_STATE` and `INTERNET` permissions and without explicitly requesting a geolocation related permission.

Table 3.4: Servers where Wi-Fi related information is sent by 88 dynamically analyzed applications.

Info	Third-parties	First-parties	# Apps affected
MAC Address	appsflyer.com (SSL), revmob.com (SSL), adsmogo.mobi (plain-text), adsmogo.org (plain-text), vungle.com (plain-text), supersonicads.com (plain-text), trademob.net (SSL), sponsorpay.com (SSL), beintoo.com (SSL), adsmogo.com (plain-text), 115.182.31.2/3/4 (plain-text)[7], tapjoyads.com (SSL)	Not found	13
(B)SSID of connected AP	inmobi.com (SSL), 93.184.219.82 (plain-text)	Not found	2
Wi-Fi Scan Info	inmobi.com (SSL), fastly.net (SSL)	badoo.com (SSL), foursquare.com (SSL)	5

3.4.2 Dynamic analysis

In Section 3.3 we speculated that applications can infer PII using the methods made available with the `ACCESS_WIFI_STATE` permission. We now want to confirm if these applications (or third-party libraries they embed) actually send private information to remote servers. We performed this dynamic analysis on 88 applications that access, at least, two privacy sensitive methods. For this purpose, we used MobileAppScrutinator.

Table 3.4 presents the list of servers to which PII obtained with the `ACCESS_WIFI_STATE` permission was sent. A number of third-parties present inside these applications are collecting the Wi-Fi MAC address and send it to their servers (sometimes in clear). Accessing Wi-Fi MAC address is really serious as it is a hardware-tied unique identifier that remains the same all along the lifespan of the device and can be used to tie both on-line and physical profile of a user (see Section 3.3). Looking at this list of servers in the Table 3.4 where Wi-Fi MAC address is sent, most of them belong to A&A companies. This clearly suggests that those actors use the MAC Address as a unique identifier to track users.

Also, both first (`Badoo.com`) and third-parties (`inmobi.com`) collect the SSID and BSSID of the AP to which the device is connected. Such a database of users and their Wi-Fi APs can easily reveal various relationships between users: a lot of information can be derived on the social relationships between users based on type of Wi-Fi APs to which users are connect to. For example, a protected Wi-Fi at home/work or the time/location at which two users connect to reveal close connections between them.

We found that Badoo and Foursquare applications send the list of surrounding Wi-Fi APs (SSIDs, BSSIDs, signal strength, etc.) to their respective servers. However, both applications have `ACCESS_FINE_LOCATION` permission and can get precise device geolocation from the regular Android APIs.

We even found some third-parties (e.g., `inmobi.com` and `fastly.net`) sending the list of surrounding Wi-Fi APs to their servers, and they are present inside various applications. Focusing on the communication inside various applications to `inmobi.com` server, we find that `inmobi.com` library works in two modes: if it is included in an application having `ACCESS_FINE_LOCATION`, it accesses the fine-grained geolocation retrieved by the system

[7]The Wi-Fi MAC address is hashed (SHA-1) before being sent over the network in clear-text.

along with nearby Wi-Fi APs (possibly to enrich their own database); otherwise, if the application doesn't have this permission, it derives device geolocation by querying their geolocation server with the list of surrounding Wi-Fi APs. As an example, code from `inmobi.com` inside SimSimi (com.ismaker.android.simsimi) application sends the list of surrounding Wi-Fi APs to its server to derive device geolocation, because this application has neither the `ACCESS_FINE_LOCATION` nor `ACCESS_COARSE_LOCATION` permissions.

Finally, we didn't encounter any application sending Wi-Fi configuration information over the network (which is good for privacy) but this might be the case in near future. Also it might be possible that our dynamic analysis technique could not detect PII leakage in case applications employ custom data modification methods.

3.5 User perception

Sections 3.3 and 3.4 have respectively demonstrated the potential privacy threats and the actual situation today on Google Play. In this section we study how users perceive the `ACCESS_WIFI_STATE` permission. More precisely we conducted an on-line survey involving 156 Android users and we studied their perception of the privacy risks associated with this permission. We show that Android permissions are often misunderstood by users who do not necessarily understand their privacy implications [109].

3.5.1 Survey description

Our survey has been performed with Google Docs and diffused through social media and multiple mailing-lists. It was composed of 12 questions divided into 3 parts:

- the first part focuses on demographic information such as age, gender and professional category;
- the second part is about user attitude towards privacy and user's experience in using the Android system;
- the third part evaluates user's perception of the relative privacy risks associated with several permissions, and in particular how users understand the implications of the `ACCESS_WIFI_STATE` permission.[8]

The third part of the survey starts with a series of questions where the respondent must evaluate the privacy risks associated with 5 selected Android permissions on a scale of 1 to 10. Along with `ACCESS_WIFI_STATE` permission, we selected `CHANGE_WIFI_STATE` and `ACCESS_NETWORK_STATE` permissions in the '`Network Communications`' group to understand how the user differentiates permissions belonging to the same group but giving access to different type of network-related data. We also selected `ACCESS_FINE_LOCATION` that is the permission explicitly required by applications to get device geolocation. As a device can also be geolocalized indirectly by applications having the `ACCESS_WIFI_STATE` permission, the `ACCESS_FINE_LOCATION` permission is selected in order to compare how users evaluate the privacy risks of both permissions. Finally, the `READ_CONTACTS` permission is selected as a reference since the name clearly indicates the associated privacy risks.

One might argue that the geolocation information obtained using Wi-Fi APs might not be as accurate as the geolocalization obtained through GPS with the `ACCESS_FINE_LOCATION` permission. However Wi-Fi based geolocation can be as accurate as GPS in urban scenarios [138, 78]. In addition, contrary to GPS, the Wi-Fi based geolocation can be used both indoors and when a user turns the GPS off to save battery.

[8] The permission were presented using a screenshot of the permission's description (as showed to the user by the Android system).

Table 3.5: User understanding of the `ACCESS_WIFI_STATE` permission. Correct answers are shown in green cells.

With `ACCESS_WIFI_STATE` permission and an Internet access, an application can ...	Responses		
Options	True	False	Don't know
✓ Check if the device is connected to the Internet through Wi-Fi	89.74%	5.77%	4.49%
✗ Turn the Wi-Fi on or off	6.41%	85.26%	8.33%
✗ Get the list of your contacts	6.41%	86.54%	7.05%
✓ Get the list of surrounding Wi-Fi networks	75.00%	12.18%	12.82%
✓ Get the list of configured Wi-Fi networks	65.38%	16.67%	17.95%
✗ Connect the device to a Wi-Fi network	21.79%	67.31%	10.90%
✓ Get the device location	48.08%	41.67%	10.26%
✓ Get one of the device unique identifiers	46.79%	17.31%	35.90%
✓ Get some of the previously visited locations (even before the App is installed)	35.90%	42.95%	21.15%

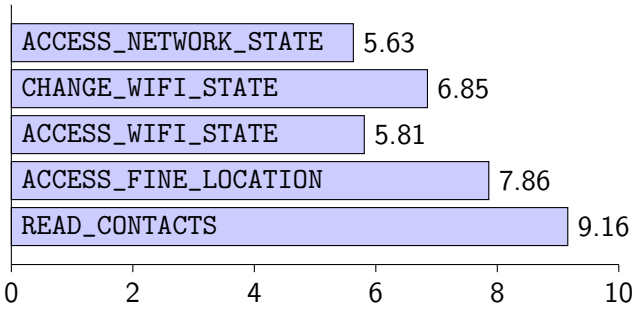


Figure 3.3: Average privacy risk rating for the considered permissions on a scale of 10.

3.5.2 Results of the survey

In total, 190 users completed the survey from February 22 to 27, 2014. We discarded responses from 34 users who never used an Android device. So the results and analysis presented below are based on the responses of 156 users who have some experience with Android.

The responses to the questions for each permission allowed us to have a comparative view of the perceived privacy risks. The average privacy risk ratings on a scale of 10 is presented in Figure 3.3. Overall, `ACCESS_FINE_LOCATION` and `READ_CONTACTS` are rated the highest for privacy risks whereas `ACCESS_NETWORK_STATE` and `ACCESS_WIFI_STATE` are rated the lowest. In particular, users rate `ACCESS_WIFI_STATE` as less risky than `ACCESS_FINE_LOCATION`. This is typically an error: not only geolocalization but also many other PII can be obtained through it (see Section 3.3). Therefore the privacy risks of this permission should have been rated higher than that of `ACCESS_FINE_LOCATION`.

The results for the question about the fine understanding of `ACCESS_WIFI_STATE` permission are presented in table 3.5. The correctness of the answers greatly varies across the questions. Thus we organised them into three groups, based on the fraction of correct answers they received.

The first group includes questions correctly answered by the majority of respondents (more than 75% of correct answers). We find questions about the basic functionalities of the `ACCESS_WIFI_STATE` permission (e.g., checking Internet connectivity through Wi-Fi and getting the list of surrounding Wi-Fi networks) as well as privileges that are not granted by the permission (e.g., turning the Wi-Fi on or off and getting the list of contacts).

The second group includes questions having received a majority of correct answers, but

fewer than for the first group (in practice more than 60%). We find questions about the ability of the application to access the list of configured networks and to connect the device to a Wi-Fi network.

Finally, the third group includes questions having received the lowest rate of correct answers (below 50%). Those questions concerns the ability of getting current or past geolocation information as well as a device unique identifier. We remark that these poorly understood capabilities are also the most privacy invasive. Even though a majority of the respondents failed to correctly answer the last set of questions, there is still a significant proportion of respondents who answered correctly (more than 35% correct answers).

3.6 Conclusion and Potential Solutions

Various studies in the past already confirmed that Android permission system is not effective at warning users about the potential danger with respect to privacy. Moreover, it is also shown that most users either do not pay attention to the permissions an app requires or do not understand the permissions during installation of apps. In this study, we confirm these findings by conducting a user survey and highlight the need for improvements in the permission system.

We find that the problem is not only the fact that users do not pay attention or do not understand the permissions but the fact that these permissions are not capable of protecting the user personal information. We presented one example of such innocuous permission but it is not the only one. Various other Android permissions are either too coarse-grained or personal information can be derived from the data accessible with the permission. Today most unique identifiers are accessible to apps without requiring a permission and even the ones which do require the permissions, are not protected by a dedicated permission. We advocate that all unique identifiers should be protected by a permission as they could be used to track device activities. In fact, a dedicated permission that clearly states its existence, should be put in place to protect abusive access of unique identifiers. Finally, the description of each permission should clearly include the data that is protected by that permission.

More concretely, this chapter presented what PII could be directly obtained or indirectly derived from data accessible to applications thanks to the `ACCESS_WIFI_STATE` permission. We showed that a large number of applications request this permission and then, with the help of an online survey, we found that users often fail to perceive privacy implications associated with this permission. Our analysis of a representative set of most popular applications in each category on Google Play revealed that a number of both first and third-parties have already started to exploit this permission to access or derive user PII.

The results of this study call for changes in the Android permission system. First of all, the access to Wi-Fi scan results should be protected with location permissions as is currently the case to access neighbouring cell towers information. Secondly, the `ACCESS_WIFI_STATE` permission description should explicitly state the various PII that can be directly obtained (e.g., MAC address that can be used for tracking) or inferred from it (e.g., travel history). Finally, the `ACCESS_WIFI_STATE` permission should be placed in the list of dangerous permissions as it is more privacy-sensitive than some of the permissions already in the list.

Chapter 4

Measuring Re-identification Risk Posed by Smartphone Apps

Contents

4.1 Introduction

People are all unique the way they are or they look. They can be easily identified from their DNA sequences, fingerprints, Iris scans, web browsers and so on. Also, a combination of various attributes, such as their age, address or religion [193] makes them unique. Recently, some studies have shown that people are also unique in the way they behave. For example, de Montjoye *et al.* illustrated this *behavioural uniqueness* by showing that people are unique in the way they move around [99]. In fact, they show that only four spatio-temporal positions are enough to uniquely identify a user 95% of the times in a dataset of one and a half million users. Similarly, other studies showed that people are unique in the way they purchase goods online [100] or configure their browser [101] or browse the web [168].

As smartphones have been widely deployed today all over the world and the list of installed/running applications (apps) on them is readily available to be accessed, the threat in terms of user privacy is huge if this data is collected, used and released without sufficient diligence in terms of privacy. This threat comes in two flavors: first, the semantics of the installed apps can tell a lot about the users' habits and interests [188], and second, the unicity of installed apps could make a user re-identifiable if this dataset is released. In fact, regarding the first privacy threat, [188] showed that user traits such as religion, relationship status, spoken languages, countries of interest, and whether or not the user is a parent of

small children, can be easily predicted from the list or even the categories of the installed apps on smartphones. In this chapter, we focus on the second privacy threat and measure the unicity of installed apps to be able to measure the risk of re-identification if app datasets are released in public or shared between two entities.

It is quite in the news these days [1] that Twitter has started to collect the list of apps that a user has installed. They claim to use this information for targeted interest-based advertising among others. However, it might be a privacy concern if Twitter shares this list of installed apps with an advertising company, even in pseudo-anonymized form, i.e., after removing all direct user identifiers (and even if app names are replaced with their hashes). This is because the advertising company might implicitly know a subset, say K, of installed apps of a user in which their ad library is present. So if K apps are enough to uniquely identify a user in the dataset, the advertiser would be able to re-identify the user in the Twitter dataset, and hence, learn about all the other installed apps of that user. By knowing this whole list of installed apps of a user, the advertiser can learn about that user's interests and habits (as demonstrated in [188]), and consequently, might be able to deliver the targeted ads directly in these apps in which its library is present. We believe that this is a real privacy threat to smartphone users (both Android and iOS) today as apps running on these OSs can access the list of installed/running apps. It is to be noted that Android apps do not require any permission to access the whole list of installed apps. On iOS, Apple does not provide a public API to access the list of installed apps but apps can get the list of currently running apps at any time. And if an app makes a frequent scan of currently running apps over a period of time, the list can converge very fast to the list of installed apps.

Contributions: The main contributions are as follows.

- We show that 99% of the lists of installed applications of users are unique out of a total of 55 thousands users. Moreover, as few as two applications are sufficient for an adversary to identify an individual's application list with a probability of 0.75 in our dataset. The re-identification probability increases to almost 0.95 if the adversary knows 4 apps. We stress that these results were obtained without considering any system apps (which are common for all users), and apps were identified only by the hash of their names. Incorporating additional information into their identifiers, such as app version, time of installation, etc., would increase these probabilities even more.

- We propose an unbiased estimate of the real uniqueness of any subset of applications, i.e., the probability that a randomly selected subset of apps with cardinality K is unique in the dataset. For this purpose, we use a Markov Chain Monte Carlo method to sample subsets of applications from a dataset uniformly at random. We prove that this chain is generally fast-mixing with most practical datasets, i.e., has a running time complexity which is roughly linear in the dataset size and K. This result might be of independent interest, as this technique can be used to sample subsets with arbitrary cardinality from any set-valued dataset.

- We attempt to predict the uniqueness of lists of applications in larger datasets using standard non-linear regression analysis. Our learned model performs well on our limited app dataset as well as on mobility data with sufficiently large number of users. However, in case of mobility data, we find that the model is not able to predict well if it is trained with smaller datasets, e.g., of the size of our app dataset. Therefore, we conclude that our app dataset at hand might probably be too small to accurately predict the uniqueness in a larger datasets such as all Android users worldwide.

[1] `http://goo.gl/00FbD0`

4.2 Unicity as a measure of re-identifiability

Let $\mathbb{A}$ denote the universe of all apps, where each application is represented by a unique identifier in $\mathbb{A}$. A dataset $D \subseteq 2^{\mathbb{A}} \setminus \{\}$ is the ensemble of all apps of some set of individuals, where $|D|$ denotes the number of individuals in D. A record D_u, which is a non-empty subset of $\mathbb{A}$, refers to all apps of an individual u in D. A set of applications with cardinality K is shortly called K-apps henceforth. The set of all K-apps over $\mathbb{A}$ is denoted as $\mathbb{A}^K$.

Definition 1 (Unicity)**.** *Let $supp(x, D)$ denote the support of $x \in \mathbb{A}^K$ in dataset D, i.e., the number of records in D which contain x. Then,*

$$H_1 = \frac{|\{x : x \in \mathbb{A}^K \wedge supp(x, D) = 1\}|}{|\{x : x \in \mathbb{A}^K \wedge supp(x, D) \geq 1\}|}$$

is defined as the unicity (or uniqueness) of K-apps in D.

The unicity of K-apps is the relative frequency of K-apps which are contained by only a single record. In general, *relative abundance distribution* (RAD)[2] is a relative frequency histogram $\mathbf{H} = (H_1, H_2, \ldots, H_n)$ of K-apps with respect to a dataset D, where H_i denotes the relative frequency of K-apps which are contained by exactly i records in D, i.e., $H_i = \frac{|\{x:x\in\mathbb{A}^K\wedge supp(x,D)=i\}|}{|\{x:x\in\mathbb{A}^K\wedge supp(x,D)\geq 1\}|}$.

Unicity is strongly related to re-identifiability, and we use it as a measure of privacy in this chapter: it is the probability that an adversary, who only knows K applications installed on a user's device, can single out the record of this user in D. Indeed, any K-apps which is unique in D can be used as a personal identifiable information (PII) of its record owner. Specifically, if the adversary knows such K-apps, it can easily identify the corresponding record and retrieve all the applications installed by its owner, even if D is pseudo-anonymized (i.e., does not contain any direct PII such as device ID or personal name). Therefore, large unicity usually indicates a serious privacy risk in practice.

4.3 Approximating unicity with sampling

To compute unicity, and RAD in general, the support of all different K-apps in D should be calculated. However, this is usually prohibitively expensive in practice. Therefore, like previous works [100, 99], we rely on sampling to estimate unicity. In particular, let Ω^K denote the set of all K-apps which occur in at least one individual's record, i.e., $\Omega^K = \{x : x \in \mathbb{A}^K \wedge supp(x, D) \geq 1\}$. We randomly sample a set V of K-apps from Ω^K, and approximate the real unicity H_1 by the sample unicity $\hat{H}_1 = \frac{|\{x:x\in V\wedge supp(x,D)=1|}{|V|}$, where $V \subseteq \Omega^K$ is the sample set, and $n = |V|$ is the sample size.

4.3.1 Biased vs. unbiased estimation of unicity

How should we sample K-apps from the dataset? A popular technique, which has been used in several works [100, 99], first samples a user uniformly at random in D, and then a set of K applications from this user's record also uniformly at random. However, this simple technique provides a *biased estimation* of the unicity in Definition 1, if the estimator remains the sample mean $\hat{H}_1$, since $E[\hat{H}_1] \neq H_1$. In fact, K-apps which occur in more records of D become more likely to be selected by this approach (assuming records have similar sizes). As a result, this sampling method is biased towards more popular K-apps, and the measured unicity is an underestimation of the real unicity H_1 what one would get with an unbiased estimator of H_1. Such an unbiased estimator can be the sample

[2]This term is often used in the field of ecology to describe the relationship between the number of observed species as a function of their observed abundance.

unicity $\hat{H}_1$ of K-apps which are sampled truly uniformly at random from D. This is also illustrated in Figure 4.3, where the sample unicity of biased and unbiased (i.e., uniform) samples are reported.

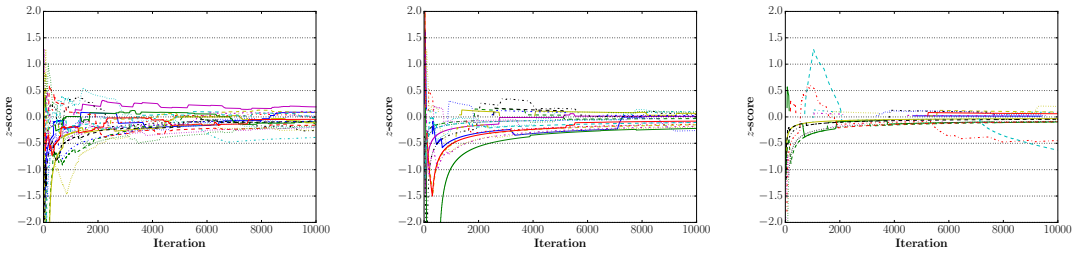
Before describing our unbiased estimation of unicity H_1, we shed some light on the privacy semantics behind the two sampling approaches. The biased technique approximates the success probability of an adversary who is more likely to know popular K-apps from the application set of any user. For instance, continuing the the case of the advertiser from Section 6.1, the advertiser's library should be more likely to be used by popular apps (such as Facebook, Twitter, etc.), which are installed on many devices, rather than by other less popular apps. However, this is not necessarily true and in general, an advertiser's library can be included in any K-apps of a user. In the rest of the chapter, we assume that the adversary can learn *any* K-apps of *any* users in D with equal probability, which is the most general assumption in practice. Therefore, we are interested in an unbiased estimator of H_1.

4.3.2 Uniform sampling of K-apps

A unbiased estimation of H_1 is obtained, if $\hat{H}_1$ is computed over a sample set where each K-apps can appear with equal probability. Hence, our task is to sample an element from Ω^K uniformly at random for any K. A first (naive) approach could be to use rejection sampling, i.e., sample a candidate K-apps from $\mathbb{A}^K$ uniformly at random, and then accept this candidate as a valid sample only if it also occurs in D. Otherwise, repeat the process until a candidate is accepted. Although sampling a candidate from $\mathbb{A}^K$ is straightforward, it is very likely to be non-existent in D (especially if K is large), and hence, its running complexity is $O(|\mathbb{A}|^K)$ in the worst case. An alternative approach could be to enumerate Ω^K, and choosing one element directly from Ω^K uniformly at random. However, the complexity of this approach is still $O(|D|(\max_u |D_u|)^K/K!)$. Unfortunately, these naive methods provide acceptable performance only if K is small. As Table 4.1 shows, in our dataset, $|\mathbb{A}| = 92210$, $\max_u |D_u| = 541$, $|D| = 54893$, and we wish to estimate the unicity when $1 \leq K \leq 10$.

We instead propose a sampling technique based on the Metropolis-Hastings algorithm [157, 88], which is a Markov Chain Monte Carlo (MCMC) method. Our proposal has a worst-case complexity of only $O(K|D|/H_1^*)$, where H_1^* is roughly the unicity of K-apps in D. As the unicity of K-apps is large, especially if K is large, the complexity is approximately $O(K|D|)$ in practice. Hence, our sampling technique remains reasonably fast even for larger values of K.

In particular, we construct an ergodic Markov chain, denoted by $\mathcal{M}$, such that its stationary distribution π is exactly the distribution that we want to sample from, that is, the uniform distribution over Ω^K. Each K-apps in Ω^K corresponds to a state of $\mathcal{M}$, and we simulate $\mathcal{M}$ until it gets close to π, at which point the current state of $\mathcal{M}$ can be considered as a sample from π. $\mathcal{M}$ is detailed in Algorithm 1. At each state transition, $\mathcal{M}$ picks a candidate next state C independently of the current state S (in Line 6-7). In Line 8, the candidate is either accepted (and $\mathcal{M}$ moves to C) or rejected with certain probability (in which case the candidate state is discarded, and $\mathcal{M}$ stays at S). The main idea is that, at each state, we use the fast but biased sampling technique, which is described in Section 4.3.1, to propose a candidate C (in Line 6-7). We correct this bias by adjusting the acceptance/rejection probability (in Line 8) accordingly; $\mathcal{M}$ is more likely to accept such K-apps which are less likely to be proposed in Line 6-7. Indeed, as $\pi(S) = \pi(C)$, the probability of acceptance is $\min\left(1, \frac{Pr[\text{“}S \text{ is proposed”}]}{Pr[\text{“}C \text{ is proposed”}]}\right) = \min\left(1, \frac{\sum_{\forall u: U_u \supseteq S} 1/\binom{|U_u|}{K}}{\sum_{\forall u: U_u \supseteq C} 1/\binom{|U_u|}{K}}\right) = \min\left(1, q(S)/q(C)\right)$. A more formal analysis is described in the section B.1 of the appendix B of this thesis. The proofs of all the theorems can be found in the section B.1 of the



(a) $K = 5$ (b) $K = 6$ (c) $K = 7$

Figure 4.1: Convergence of our Markov chain $\mathcal{M}$. The z-score, depending on the number of iterations t, of 20 independent chains are plotted.

appendix B of this thesis.

Algorithm 1 MCMC sampling ($\mathcal{M}$)

Input: Dataset D, K, # of iterations t
Output: A sample $S \in \Omega^K$
Let $U := \{D_u : |D_u| \geq K \wedge D_u \in D\}$
Let S be an arbitrary K-apps in Ω^K
for $k = 1$ **to** t **do**
    Select an individual $u \in [1, |U|]$ uniformly at random
    Select a subset $C \subseteq U_u$ uniformly at random such that $|C| = K$
    Let $S := C$ with probability $\min(1, q(S)/q(C))$, where $q(x) = \sum_{\forall u: D_u \supseteq x} \prod_{i=1}^{K} \frac{1}{|U_u|-K+i}$
return S

Theorem 1. *$\mathcal{M}$ in Algorithm 1 is an ergodic Markov chain whose unique stationary distribution is the uniform distribution over Ω^K for any K.*

Convergence of $\mathcal{M}$ How to adjust t in Algorithm 1? We prove that a "good" uniform sample from Ω^K can be obtained roughly after $O(|D|)$ iterations in most practical cases. The time that $\mathcal{M}$ takes to converge to its stationary distribution π is known as the *mixing time* of $\mathcal{M}$, and is measured in terms of the total variation distance between the distribution at time t and π.

Definition 2 (Mixing time)**.** *For $\xi > 0$, the mixing time $\tau_{\mathcal{M}}(\xi)$ of Markov chain $\mathcal{M}$ is*

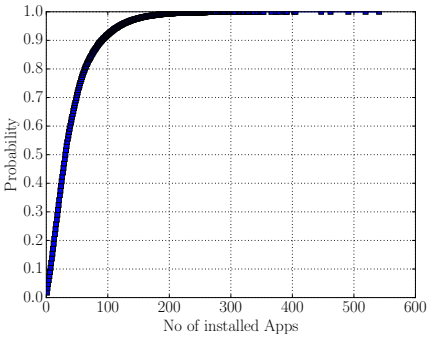
$$\tau_{\mathcal{M}}(\xi) = \min\{t' : ||P^t_{\mathcal{M}} - \pi||_{tv} \leq \xi, \forall t \geq t'\}$$

where $||P^t_{\mathcal{M}} - \pi||_{tv} = \max_{x \in \Omega^K} \frac{1}{2} \sum_{y \in \Omega^K} |P^t_{\mathcal{M}}(x, y) - \pi(y)|$ defines the total variation distance. $P^t_{\mathcal{M}}(x, y)$ denote the t-step probability of going from state x to y, and $P^t_{\mathcal{M}}$ denote the t-step probability distribution over all states.

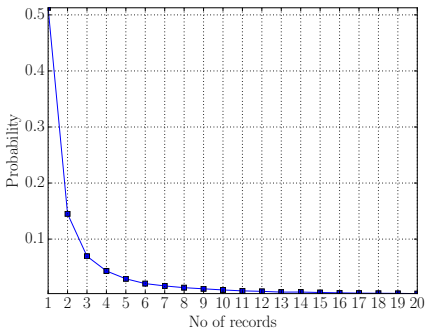
The next theorem shows that $\mathcal{M}$'s mixing time is $O(|D| \log(1/\xi)/H^*_1)$, where $|D|$ is the dataset size and H^*_1 is the unicity of K-apps from the largest record of D. As the unicity of K-apps is usually large in practice, especially if K is large, $\mathcal{M}$ is fast-mixing in general. In our dataset D, $0.6 \leq H^*_1 \leq 0.999$ for $2 \leq K \leq 9$ [3].

Theorem 2 (Mixing time of $\mathcal{M}$)**.** *Let H^*_1 denote the probability that a randomly selected set of K items from the largest record (i.e., having the most apps) in D is unique. Then, $\tau_{\mathcal{M}}(\xi) \leq |D| \ln(1/\xi)/H^*_1$ for any K.*

[3] The unicity of K-apps from a single record can easily be approximated with Inequality 4.2 using uniform samples over all K-apps from the record. Likewise the biased sampling in Section 4.3.1, this sampling is easy to implement (e.g., by choosing K items individually from the record without replacement).



(a) Cumulative distribution of installed apps.



(b) Probability distribution of number of records containing a particular app.

We emphasize that the bound in Theorem 2 is a worst-case bound, and the real convergence time can be much smaller depending on the dataset D as well as the starting state of the chain. As we show next, $\mathcal{M}$ indeed exhibits much smaller convergence time than its theoretical worst-case bound for our dataset. We detected the convergence of $\mathcal{M}$ using the Geweke diagnostic [116]; if X_t denotes a Bernoulli random variable describing whether the current state of $\mathcal{M}$ at time t is unique, and $\mathbf{X}_t = (X_1, X_2, \ldots, X_t)$, then we compute the z-score $z = \frac{E[\mathbf{X}_a] - E[\mathbf{X}_b]}{\sqrt{Var(\mathbf{X}_a) + Var(\mathbf{X}_b)}}$, where $\mathbf{X}_a$ is the prefix of $\mathbf{X}_t$ (first 10%), and $\mathbf{X}_b$ is the suffix of $\mathbf{X}_t$ (last 50%). We declare convergence when the z-score falls within $[-1, 1]$. Indeed, if $\mathbf{X}_a$ and $\mathbf{X}_b$ become identically distributed (i.e., $\mathbf{X}_a$ and $\mathbf{X}_b$ appear to be uncorrelated), the z values become normally distributed with mean 0 and variance 1 according to the law of large numbers. We simulated 20 instances of $\mathcal{M}$ each starting at different states, and plotted the z-score of each chain depending on the number of iterations t in Figure 4.1. This shows that convergence is detected roughly after 3000 steps in all chains with different values of K. When this happens, the current state can be taken as a valid sample. Hence, in the sequel, we run $\mathcal{M}$ with $t = 3000$ to obtain a uniform sample from Ω^K.

We note that q in Algorithm 1 can be computed rapidly in practice by precomputing another dataset T, where each record corresponds to an application in D, and record i contains the sorted list of all users who have application i in their record. Hence, the set of users who have a common specific K-apps can be computed easily by taking the intersection of the corresponding records in T. The complexity of this operation is $O(K|i_{\max}|)$, where $|i_{\max}|$ is the maximum record size in T, i.e., the number of users of the most popular application in D. Fast implementations of the intersection of sorted integers are described in [142].

4.3.3 Computing the sample size

In order to compute the sample size, we use the Chernoff-Hoeffding inequality [125] on the tail distribution of the sum of independent (but not necessarily identically distributed) Bernoulli random variables. In particular, if X_i denotes a Bernoulli random variable describing the event that the ith sampled K-apps is unique in D, then the deviation of the estimator $\hat{H}_1 = \sum_{i=1}^{n} X_i/n$ from $E[\hat{H}_1] = H_1$ is given by $Pr\left[\left|\hat{H}_1 - H_1\right| \geq \varepsilon\right] \leq 2e^{-2n\varepsilon^2}$, or equivalently,

$$Pr\left[\left|\hat{H}_1 - H_1\right| < \varepsilon\right] \geq 1 - 2e^{-2n\varepsilon^2} \tag{4.1}$$

where ε is the sampling error and $\sigma = 1 - 2e^{-2n\varepsilon^2}$ is the confidence. Hence, we obtain that

$$n \geq \frac{1}{2\varepsilon^2} \ln\left(\frac{2}{1-\sigma}\right) \tag{4.2}$$

Dataset size $\|D\|$	$54,893$
# of all apps in D	$92,210$
Maximum record size $\max_u \|D_u\|$	541
Minimum record size $\min_u \|D_u\|$	1
Average record size	42
Std.dev of record size	39

Table 4.1: Characteristics of our dataset D

For example, if $\varepsilon = 0.01$ and $\sigma = 0.99$, we need to sample at least 26492 K-apps from D (with replacement) to guarantee that $|\hat{H}_1 - \hat{H}_1| < 0.01$ with probability at least 0.99.

Considering RAD, suppose we aim at approximating the first k relative frequency values of $\mathbf{H}$, i.e., $(H_1, H_2, \ldots, H_k)$. Therefore, we wish to simultaneously satisfy Inequality 4.1 for each H_i $(1 \leq i \leq k)$, where $\hat{H}_i = \sum_{j=1}^{n} X'_j/n$, and $X'_j = 1$ if the jth sampled K-apps occurs in exactly i records of D, otherwise $X'_j = 0$. Hence,

$$Pr\left[\bigwedge_{i=1}^{k}\left|\hat{H}_i - H_i\right| < \varepsilon\right] \geq 1 - \sum_{i=1}^{k} Pr\left[\left|\hat{H}_i - H_i\right| \geq \varepsilon\right]$$
$$\geq 1 - 2ke^{-2n\varepsilon^2}$$

where $\delta = 1 - 2ke^{-2n\varepsilon^2}$ is the confidence. Therefore,

$$n \geq \frac{1}{2\varepsilon^2} \ln\left(\frac{2k}{1-\sigma}\right) \tag{4.3}$$

For instance, for $\varepsilon = 0.01$, $\sigma = 0.99$, and $k = 10$, we need to sample at least 38005 K-apps from D (with replacement). This will guarantee that $|\hat{H}_i - \hat{H}_i| < 0.01$ for all $1 \leq i \leq k$ with probability at least 0.99.

4.4 Evaluation

4.4.1 Dataset characteristics

The analyzed dataset comes from the Carat research project [170]. The dataset includes data from $54,893$ Carat Android users between March 11, 2013 and October 15, 2013 [198]. During this period, the Carat app[4] was collecting the list of running apps (and not the list of all installed apps) on users' devices when the battery level changes. *As collecting the list of running apps multiple times over more than 7 months is likely to sum up to the set of all installed apps of a user, we consider a record as the set of installed applications in this chapter, even if a record might not be the complete set of installed apps all the time.*

We removed system apps from all records because they are common to all users. Without system apps, our analyzed dataset contains 92,210 different applications whereas total number of apps available on the GooglePlay were around 1 million during this time[5]. Furthermore, the average number of apps installed per user in our dataset is 42 with a standard deviation of 39. Table 4.1 summarizes the main characteristics of our dataset D.

Figure 4.2a depicts the cumulative distribution of the number of apps installed by a particular user. We note that more than 90% of users have 100 or fewer applications. Probability distribution of the number of users who installed a particular app is depicted by Figure 4.2b. Notice that more than half of the apps are contained by only a single record in D.

[4] `http://carat.cs.helsinki.fi`

[5] `http://en.wikipedia.org/wiki/Google_Play`

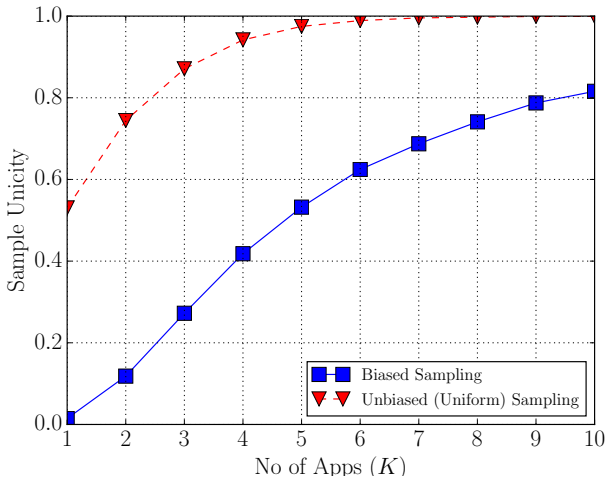


Figure 4.3: Uniqueness probability as a function of K for biased and unbiased sampling

Ethical Considerations The data were collected with the users' consent, and they were explicitly informed that their data could be used and shared for various research projects. In fact, the Carat privacy policy (available at `http://carat.cs.helsinki.fi`) clearly specifies that "Carat is a research project, so we reserve the right to publish our results online and in academic publications. We also reserve the right to release the data sets into the public domain." Also, the dataset was shared with us by the Carat team in a pseudo-anonymised form. In particular, identifiers were removed, and each application name was replaced with its SHA1 hash. It contained 54,893 records [198], i.e. one record per user. Each record is composed of the list of applications installed by the user. Furthermore, the data sharing agreement that we signed, stipulated that we cannot use the dataset to deanonymize the users in the dataset.

4.4.2 Results

We find that 98.93% of users have unique set of installed apps in D, i.e., there does not exist any other user with the same set of installed apps. This means that if we know the list of all the installed apps of a user in the dataset, we can identify that user in the dataset with a probability of 0.99. As the adversary might not always be aware of all the installed apps of a user in practice, we measure the unicity of K-apps for different values of K using our dataset D.

Figure 4.3 gives the unicity of K-apps with different values of K (changing from 1 to 10) for the two different types of sampling techniques described in Section 4.3: the biased sampling from [99, 100] and our unbiased, uniform sampling described in Section 4.3.2. In each case, we computed the sample size using Inequality 4.1 with maximum sampling error $\varepsilon = 0.01$ and confidence $\sigma = 0.99$. Otherwise stated explicitly, we use this sample size in the sequel. This results in 26492 samples for each value of K. As biased sampling favours more popular K-apps, the sample unicity $\hat{H}_1$ is less than with our unbiased approach. In particular, the difference can be as large as 0.5 for smaller values of K, while it decreases as K increases. For the unbiased estimation, the sample unicity is 0.75 with $K = 2$, and it reaches 0.99 when $K = 6$.

Figure 4.3 shows that the unicity of any K-apps is large and hence there would be a real privacy threat if such dataset was released. Moreover, Figure 4.4 depicts the relative abundance distribution in D, when $1 \leq K \leq 8$. RAD provides complementary information about users' privacy in D. In particular, even if the adversary cannot single out the record

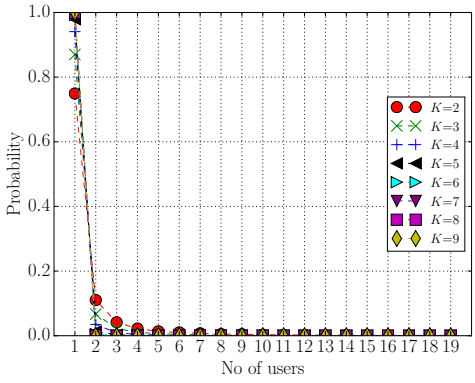


Figure 4.4: Relative abundance distribution of apps for different sizes of sets of apps.

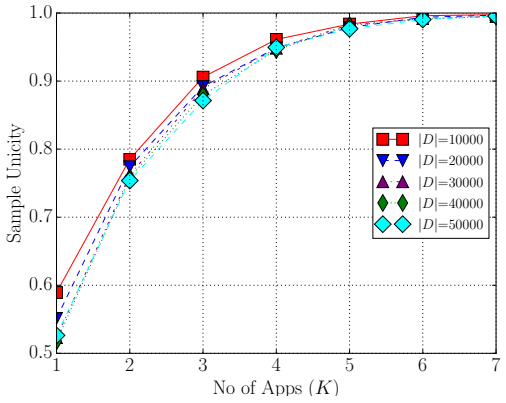


Figure 4.5: Effect of number of users.

of the target user in D, it might still learn new information about him/her. For example, if the known K-apps of the target user are shared by multiple users in D and all these users have some identical apps besides the known K-apps, then the adversary learns that the target user also has these apps installed on his/her phone. This attack is often referred to as the homogeneity attack in the literature [150]. We computed the required sample size using Inequality 4.3 with $\varepsilon = 0.01$ and $\sigma = 0.99$ for $k = 20$. This gives 41470 samples overall, which were taken with our uniform sampler $\mathcal{M}$.

To study the effect of number of users on unicity, we randomly select subsets of users of different sizes from D, and calculate the sample unicity within these subsets. Figure 4.5 depicts how unicity changes with the number of users in our dataset. We find that unicity decreases if the user number increases. However, this decrease becomes less significant for larger number of users. This is probably due to the fact that the number of apps starts to saturate if the user number increases.

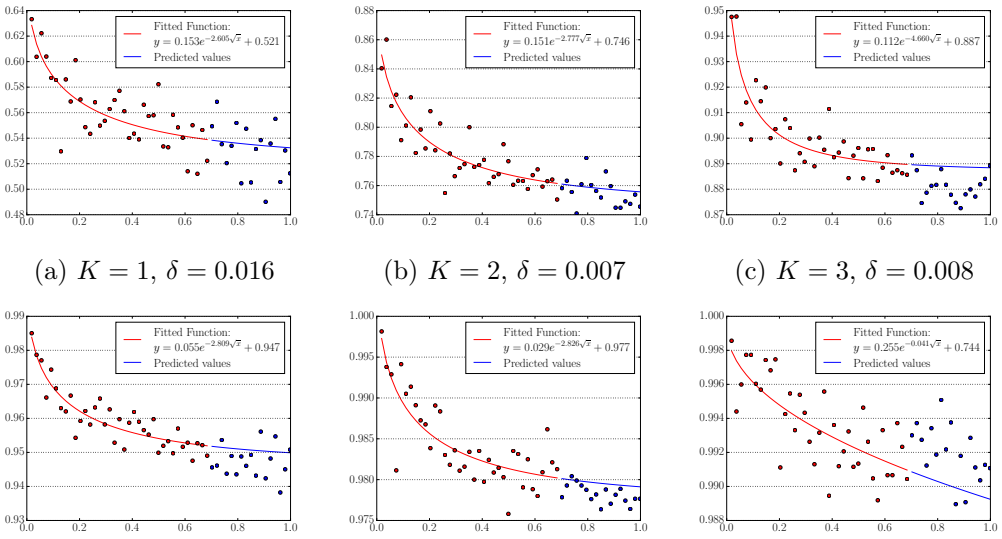
As the size of our dataset is much less than the population size of all Android users worldwide (which was roughly 1 billion as of 2014 [6] with 1.2 million different applications available on GooglePlay[7]), we aim at predicting the unicity in a larger dataset (possibly in the whole population) in the next section.

4.5 Unicity Generalization for larger datasets

Information surprisal can be used to measure uniqueness in the population D [101]. In our case, the population is all the Android users worldwide, to which we want to generalize our results. As information surprisal of any K-apps $\{A_1, A_2, \ldots, A_K\}$ over D is equal to $-\log(Pr[A_1, A_2, \ldots, A_K])$, we must need to first measure the co-occurence probability $Pr[A_1, A_2, \ldots, A_K]$ of these apps in D. The co-occurence probability can be easily computed if we can assume that apps co-occur independently in the dataset as $Pr[A_1, A_2, \ldots, A_K] \approx \prod_{i=1}^{K} Pr[A_i]$, and $Pr[A_i]$ (the popularity of app A_i) can be obtained from the download count of A_i available on Google PlayStore. However, this is not the case in a real-world scenario as there exist correlation between apps installed by a user. As our dataset is very likely to be too limited to capture this correlation (as our dataset contain only 93K distinct apps whereas there are more than 1.2 million available apps on GooglePlay), we cannot take this approach to measure the uniqueness in the population of Android users. We rather employ regression analysis on our dataset which does not rely on this correlation information to predict the unicity in a larger dataset.

[6] http://www.engadget.com/2014/06/25/google-io-2014-by-the-numbers/

[7] http://www.appbrain.com/stats/number-of-android-apps



(a) $K = 1$, $\delta = 0.016$ (b) $K = 2$, $\delta = 0.007$ (c) $K = 3$, $\delta = 0.008$

(d) $K = 4$, $\delta = 0.005$ (e) $K = 5$, $\delta = 0.001$ (f) $K = 6$, $\delta = 0.002$

Figure 4.6: Unicity generalization for different values of K, trained all with maximum 37000 users. The learnt models (i.e., $f(x)$) are present in the legend. x-axis corresponds to normalized dataset sizes with a normalization factor of 1/54893, and y-axis depicts sample unicity.

For regression analysis, we randomly create datasets of different sizes from our original dataset and compute the sample unicity for these datasets of different sizes. This gives us the tuples (x,y) where x is the number of users in a particular dataset (independent variable) and y is the calculated unicity value dependent on x. Here, we assume that the unicity value y only depends on the number of users x. We must note that, in reality, unicity depends on many factors such as the characteristics of the users, how many (un)popular applications users tend to have, etc. As it is difficult to take into account all these factors either because they are unknown or hard to measure, we assume that unicity in general is a "proper" function of only the total number of users in the dataset. That is, all other dependent factors are implicitly incorporated into the model, i.e., the general form of the function.

Once we have these (x,y) tuples, our goal is to select the best model and its parameters that capture the relation between x and y. The overall approach is as follows: we divide our (x,y) tuples in training and test sets. We select the best model (i.e., a function family) based on the general characteristics of application unicity and then learn its exact parameters using our training set. Finally, using the best model thus obtained, we test its accuracy on the test set. This model should be able to predict the unicity value for any dataset of arbitrary size.

Training and Testing We divide our original dataset in 54 smaller datasets, each of size varying from 1k to 54k. We take the first 70% of all (x,y) points for training and the last 30% (corresponding to larger datasets) for testing. We deliberately take the last points corresponding to larger datasets for testing set because we aim to evaluate our model performance on larger datasets, i.e., we want to test how accurately the learned model could be extrapolated.

As we divided our datasets by randomly selecting users out of the original dataset, users in the training and testing set may overlap. However, we found that unicity merely depends on the number of users in the dataset and not specifically on the underlying individuals.

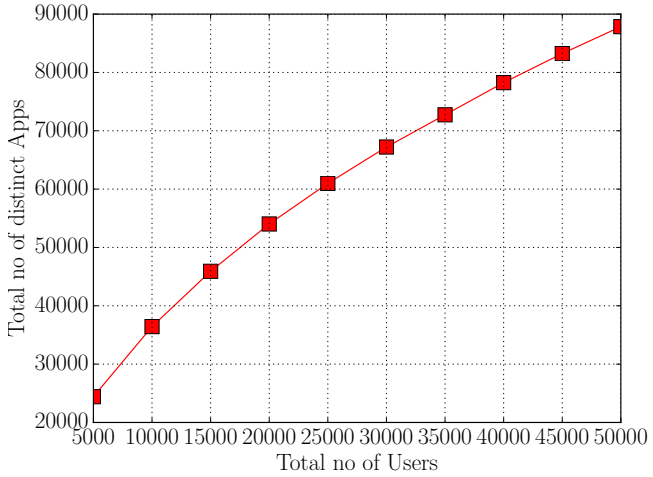


Figure 4.7: No of distinct apps installed by users

For example, we computed the unicity of 50 different sets of 1000 users selected randomly, and found out that the variance of the measured sample unicity is very small.

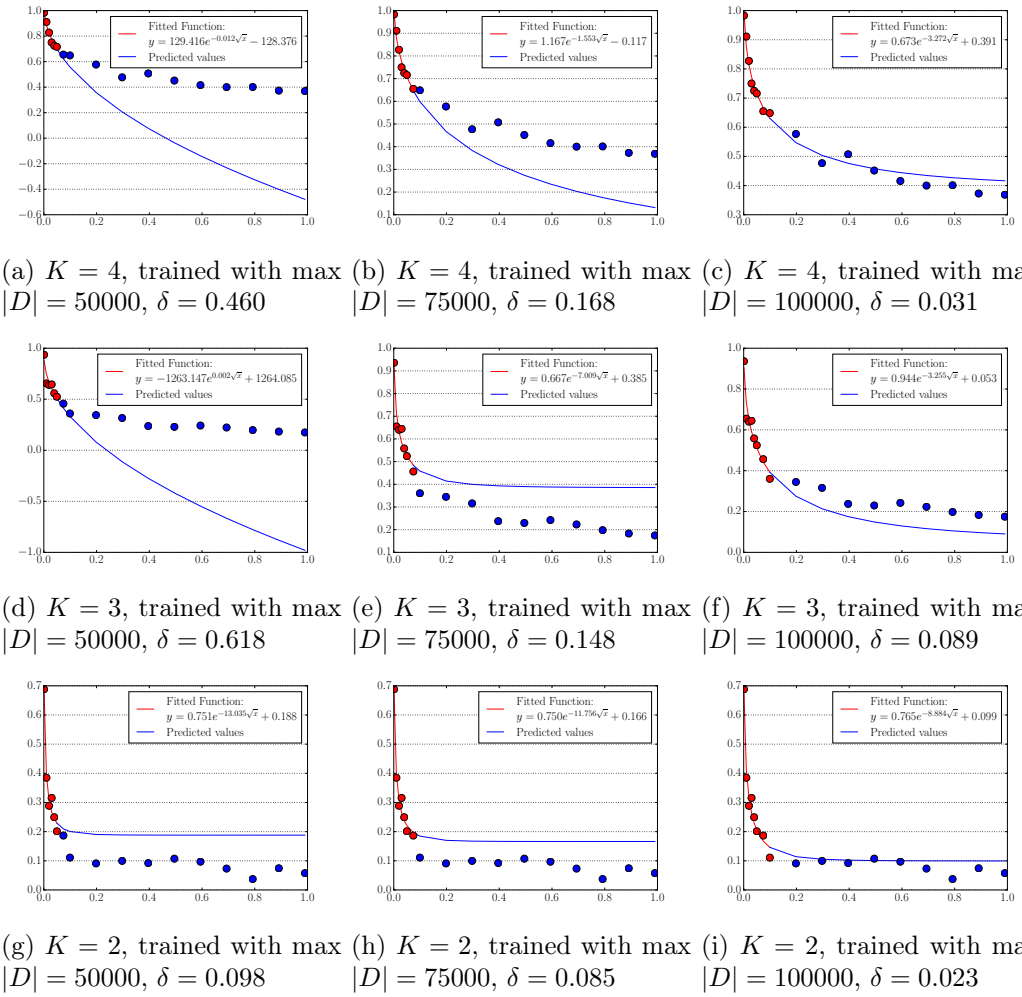
Model selection To select our model, we first tried linear regression with non-linear basis functions (polynomials of various orders) with and without regularization. However, they provided very inaccurate predictions of unicity. Finally, we selected the following exponential model describing an exponential decay of unicity:

$$f(x) = a \cdot \exp(-b\sqrt{x}) + c \tag{4.4}$$

The rationale behind choosing this model is as follows. Figure 4.7 shows that if additional users were added to our dataset, the number of apps would reach the maximum number of apps in the population early as there are fewer apps on GooglePlay than total number of Android users. This suggests that, after a certain point, additional users would not bring many new apps but still, they would bring new combinations of already existing apps. The addition of new combinations of apps should lead to the increase in unicity. However, the newly added users can lead to the decrease in unicity as well due to the fact that they can also have many already existing combinations of apps. As these two effects of adding new users to the dataset run opposite to each other, we suppose that unicity converges to a value greater than zero which is denoted by c in Equation 4.4. Indeed, as Figure 4.5 shows, although unicity decreases with the increase in the user number, the amount of this decrease tends to decrease as well. A similar observation was made in [99]. Also, we used square root of x in the exponent in Equation 4.4 because taking square root is variance-stabilizing[8]. In fact, we tried other powers of x in the exponent but square root lead to the best results.

The goal of the regression is to compute parameters a,b and c in Equation 4.4 from the training set (x,y) tuples. In fact, these parameters might be computed employing either standard non-linear regression directly or by first transforming Equation 4.4 into linear form and then applying linear regression. We use standard non-linear regression because it explicitly computes the lower bound on the unicity value (i.e., c in Formula 4.4). The value of x is normalized[9] by dividing x with the maximum size of the dataset for which we want to predict the unicity value.

[8] `https://en.wikipedia.org/wiki/Variance-stabilizing_transformation`
[9] `https://en.wikipedia.org/wiki/Feature_scaling`



(a) $K = 4$, trained with max $|D| = 50000$, $\delta = 0.460$ (b) $K = 4$, trained with max $|D| = 75000$, $\delta = 0.168$ (c) $K = 4$, trained with max $|D| = 100000$, $\delta = 0.031$

(d) $K = 3$, trained with max $|D| = 50000$, $\delta = 0.618$ (e) $K = 3$, trained with max $|D| = 75000$, $\delta = 0.148$ (f) $K = 3$, trained with max $|D| = 100000$, $\delta = 0.089$

(g) $K = 2$, trained with max $|D| = 50000$, $\delta = 0.098$ (h) $K = 2$, trained with max $|D| = 75000$, $\delta = 0.085$ (i) $K = 2$, trained with max $|D| = 100000$, $\delta = 0.023$

Figure 4.8: Unicity generalization for different values of K for location data, trained with datasets of different sizes. The learnt models (i.e., $f(x)$) are present in the legend. x-axis corresponds to normalized dataset sizes with a normalization factor of $1/10^6$, and y-axis shows sample unicity.

Results As an error metric, we measure the average absolute error denoted by δ, i.e.,

$$\delta = (1/n) \sum_{i}^{n} |y_i - f(x_i)|$$

where n is the number of predicted points, y_i is the real unicity value, and $f(x_i)$ is the predicted value.

Figure 4.6 presents how our exponential model performs on the test set for different values of K. Although our model can predict the trend of the unicity for large number of users, it slightly overestimates the real unicity in the test set. Nevertheless, the average error δ on the test set is only around 0.01. As our app dataset is very small as compared to the whole Android population, we cannot evaluate performance of our model for large number of users, e.g., a few million, or the whole Android population. Therefore, we cannot claim that our model will be able to accurately predict the unicity for datasets having large number of users even if it performs reasonably well on our test data.

Model validation on a different dataset To further demonstrate that our model is a meaningful approach to predict unicity in large populations, we test it on a large mobility

dataset provided by a telecom operator in Europe. This dataset contains the Call Data Records (CDR) of 1 million users from a large european city over 6 weeks. Each record in the dataset corresponds to a user and contains the set of his/her visited cell towers, where the total number of different cell towers is 1303. From this dataset, we created smaller datasets of different sizes (x ranging from 1000 to 1 million users). Then, we trained our model on the first 6 points (i.e., until the dataset size of $50,000$). Figure 4.8 shows that the model does not predict accurately the unicity for large datasets and the error can be as large as 0.6. Next, we trained the model on the first 7 points (i.e., until the dataset size of $75,000$). In this case, we find that the model performs significantly better than in the previous case with an error of 0.13 on average. Finally, we trained the model on the first 8 points (i.e., until the dataset size of $100,000$). In this case, we find that the model have accurate predictions for larger datasets, e.g., for a test dataset of size 1 million (10 times more than the maximum size of the dataset used in the training phase), and the error is 0.05 on average.

We find that a mobility dataset of 0.1 million users is sufficient to learn an accurate model and predict the unicity values for a larger mobility dataset. However, as we saw earlier, the model is not able to predict well the unicity of a large population if it is trained on a dataset of only $50,000$ users. This may suggest that $50,000$ users might be too small in general to learn an accurate model and therefore, our app dataset of $50,000$ users might not be sufficient to learn the model. On the other hand, even if this model performs well on a mobility dataset with 1 million users, it does not necessarily imply its good performance on large application datasets due to the different data and user characteristics. Nevertheless, these two experiments together show that our exponential model can be a meaningful approach to predict unicity in large populations.

4.6 Related work

In almost the same way as our DNA or fingerprints, it has been shown that combination of ZIP code, date of birth, and gender is likely to identify an individual uniquely [192]. The arithmetic behind arriving to this conclusion is simple [1]: all we need is 33 bits of information to uniquely identify a person among seven billion people in the world. Brian Hayes, in his article "Uniquely Me!" summarises very well how unique we are and what information can be used to uniquely identify us [124].

In [99], the authors quantify the uniqueness of human mobility traces. They study the list of visited locations at the geographic granularity of area covered by GSM cell towers and temporal granularity of one hour. The studied dataset contained 1.5 million individuals and was collected for a period of 15 months. They find that four spatio-temporal points, in this setting, are enough to uniquely identify a user 95% of the times. To check the effects of resolution, they coarsen the data both spatially and temporally. They find that the uniqueness of mobility traces decays approximately as the $1/10$ power of their resolution. Hence, they conclude that even coarse datasets provide little anonymity.

In [100], authors study three months of credit card records for 1.1 million people and show that four spatiotemporal points are enough to uniquely re-identify 90% of individuals. They further show that knowing the price of a transaction increases the risk of re-identification by 22%, on average. Finally, they show that even data sets that provide coarse information at any or all of the dimensions provide little anonymity and that women can be re-identified more than men in credit card metadata.

In [168], the authors analyse the collected browser history profiles from consenting volunteers. They find that, out of $368,284$ web histories, more than 69% of users have a unique fingerprint. Moreover, if they only consider the users that had at least four sites in their browsing history, they find that the such browsing histories were unique 97% of

times.

Another study [101] investigates how unique are our web browsers, i.e., the way the web browsers are configured. From the browser configuration of users who visited their test site, they find that the distribution of their fingerprint contains at least 18.1 bits of entropy. This means that if we pick a browser at random, at best we expect that only one in 286,777 other browsers will share its fingerprint. Among browsers that support Flash or Java, the situation is even worse, with the average browser carrying at least 18.8 bits of identifying information. 94.2% of browsers with Flash or Java were unique in our sample.

4.7 Conclusion

This chapter showed that the list of installed applications is quite unique and therefore, can be used as a unique identifier. This result has few implications on user's privacy. First, since this metadata is unique, it could easily be used to profile users, e.g., based on the category of installed apps. This is what Twitter is doing to provide interest-based targeted ads to users. Second, as a combination of even small number of installed apps is quite unique, this information could be used to re-identify users in a dataset. For example, if Twitter decided to publish the list of apps installed by its users on their smartphones, it would be easy for anyone, who knows 4 or 5 apps of a given user, to re-identify him and discover other apps that are also installed on his smartphone. This makes anonymization of this information challenging, and this is part of our future work.

As smartphones have been widely deployed/adopted today all over the world and the list of installed/running applications (apps) on them is readily available to be accesssed by apps (or included third-party libraries), the threat in terms of user privay is huge. In general, mobile users reveal many pieces of information that, when combined together, provide a lot of information about users and can be used to build personalized profiles. For example, by combining the locations and the list of apps of a user, a service, such as Twitter, can uniquely categorize a user. Since people are unique in many different known and unknown ways, preserving the privacy of mobile users is very challenging. New protection measures need to be devised.

Part II

Unveiling and Controlling Tracking on the Web

Preface

Approach / Ecosystem	Solving privacy problems at the source	Empowering users
Web	OpenWPM [106], Personal data worth [169], Leaking sensors API [166]	LightBeam [15], Ghostery [11], Disconnect [8], Privacy Badger [21], AdBlock [122], AdBlock Plus [2]

Table 4.2: The table gives some notable projects taking two approaches: : solving privacy problems at the source and empowering users. This list of projects is not exhaustive. In this thesis, we take the second approach, i.e., empowering end user by providing them with transparency and control. The related projects taking this approach are colored in gray.

In the last few years, as a result of the proliferation of intrusive and privacy-invading ads, the use of ad-blockers and anti-tracking tools have become widespread. As of the second quarter of this year, 16% of online Americans, about 45 million people, had installed ad-blocking software, according to PageFair 2015 report [49]. Meanwhile, 77 millions Europeans are blocking ads. All this accounts globally for $21,8 billion worth of blocked ads. The Internet economy is in danger since ads fuel the free content and services over Internet.

We believe that Ad blockers are only a short-term solution, and that better tools are necessary to solve this problem in the long term. Most users are not against ads and are actually willing to accept some ads to help websites. However, users want more control and transparency about the way they are tracked and profiled on the Internet, and the ads that they want to receive.

As opposed to existing ad blockers that take a binary approach, i.e., block everything if you install them or block nothing otherwise, we aim to provide users with the right tools that allow users to make fine-grained choices about their privacy and the ads that they want to receive. These tools should allow users to choose on which sites (more specifically, on which categories of sites) they want to block the trackers. For example, a user can choose to block the trackers on sites related to health or religion, but may choose not to block the trackers on sites related to sports or news. Similarly, a user might want to block ads that are targeted on some categories that he considers sensitive.

As trust between users and online entities is the key here, we provide users with transparency, e.g., what trackers are present on a website, the category (such as sports, health) of the website, if an ad is retargeted or is delivered to users based on their interests. The ultimate goal is to enforce the user choices while sustaining the ad economy of the Internet. And thanks to this transparency-enhancing feature, users can be aware of what is going on with their browsing data, and therefore can make an informed decision.

To tackle aforementioned issues, two approaches exist as shown in Table 4.2. The first approach is doing measurement studies in the lab and therefore, revealing the different tracking techniques used by A&A companies. The second approach consists in empowering

users, i.e., providing them with more transparency and control over their data. Table 4.2 presents the related projects taking these two approaches.

Below we give details of some projects that took the first approach and tried to conduct some measurements studies in the lab to bring more transparency and control over tracking on the Web.

OpenWPM is an automated platform for large-scale web measurement [106]. It has already been used by a lot of research studies to bring transparency to the Web [180, 112, 135, 59, 107, 191]. The creators of this tool have also used it to do a measurement study of 1-millions websites [105]. Thanks to their study, a number of tracking techniques have been reported to the wide audience and therefore, leading to a positive impact to the whole ecosystem.

Another line of work constitutes showing users the worth of their personal data on the Web [169]. Authors provide an analysis of the value of users' private data from the advertisers' perspective based on prices they paid for serving ads to users. They analyse how such factors as the visiting site, time, user's physical location and profile affect prices actually paid by advertisers. Their study concludes that the worth of user personal data is far less than what users expect it to be [84].

There has been some research on privacy issues related to sensors API [166] in HTML5. The report investigating battery API [167] finds that it is possible to track users for short time intervals. This is because of the fact that the API returns the batter status with a high precision. The problem can be addressed by simply returning the status of battery with lower precision. In [23], the privacy risks posed by vibration API are discussed more in detail and implementers of this API are warned to keep in mind that a potential user identifier could be derived from the use of this API.

Below are the tools that are provided to end users to bring more transparency and control over tracking and advertising on the Web. They take the second approach; as is taken by this thesis. Therefore, these are the most important related work to our work in this thesis. The cell corresponding to these projects in Table 4.2 is colored in gray.

To show users the number (and type) of trackers, usually hidden, present on a web page, various tools exist. LightBeam [15], Ghostery [11], Disconnect [8] and Privacy Badger [21] are some examples of such tools. Among these tools, LightBeam is solely for transparency purposes while others simultaneously also let users block the trackers if they want. In essence, these tools also work as tracker blocker.

In addition to these tools, there also exists tools, e.g., Ad Block [122], Ad Block Plus [2], that are specifically designed to block ads. As a result of blocking ads, such tools also block most trackers (but not necessarily all the trackers). Indeed, those trackers that do not deliver ads and are only present to track user visits across various domains, are not blocked.

Chapter 5

MyAdChoices: Bringing Transparency and Control to Advertising

Contents

5.1 Introduction

Recent years have witnessed the rise of a myriad of ad-blocking tools whose primary aim is to return control to users over advertising. In essence, ad blockers monitor all network connections that may be initiated when the browser loads a page, and prevent those which

are made with third parties[1] and may correspond to ads. To this end, ad blockers rely on blacklists manually maintained by their developing companies and, in some cases, by user communities.

Apart from the controversy stirred by the use of such lists —especially after the revelation that Adblock Plus [2], the most popular of these technologies, was getting money from ad companies to whitelist them [90]—, the main problem with these tools is that they were conceived without considering two key points: first, the crucial role of online advertising as the major sustainer of the Internet "free" services; and secondly, the social and economic benefit of non-intrusive and rational advertising. While ad-blockers might constitute a first attempt in this bid to regain control over advertising, they are extremely limited and radical in their approach: users can only choose either to block or allow all the ads blacklisted by the ad-blocking companies.

In a half-hearted attempt to address the aforementioned privacy and usability concerns, the Internet advertising industry and the World Wide Web Consortium have participated in two self-regulatory initiatives, Your Online Choices [34] and Do Not Track (DNT) [50]. Although these two initiatives make opt-out easier for users —the former to stop receiving ads tailored to their Web-browsing interests, and the latter to stop being tracked through third-party cookies—, the fact is that users have no control over whether or not their advertising and tracking preferences are honored.

With around 200 million people worldwide regularly using ad blockers[2], as well as with Apple's recent support for the development of such tools in its new iOS release [162], the economic model underlying the Web is at serious risk [171]. This has spurred a heated debate about the ethics of these technologies and the need for a solution that strikes a better balance among the Internet's dominant business model, user privacy and Web usability [70, 98, 194].

We believe that the solution necessarily implies giving users *real* control over advertising, and that this can only be achieved through technologies that enforce their actual preferences, and not the radical, binary choices provided by the current ad blockers. As a matter of fact, according to a recent survey, two out of three ad-blocker users are not against ads and would accept the trade-off that comes with the "free" content [38]; this is provided that advertising is a transparent process and they have control over the personal information that is collected [182]. Trust, through transparency, seems to be key in this regard [159]. However, because different users may have different motivations, we require tools that allow for such different choices regarding ad blocking.

5.1.1 Contribution and Plan of this Chapter

In this work, we investigate a smart Web technology that can bring transparency to online advertising and help users enforce their own choices over ads. The technology proposed in this chapter has been contrived within the project *MyRealOnlineChoices*, and aims at providing ad transparency on the one hand, and ad-blocking functionalities on the other.

The main goal of this tool is, first, to let users know what is happening behind the scenes with their Web-browsing data; and secondly, to enable them to react accordingly, in a flexible and non-radical way, by giving them fine-grained control over advertising. Its ultimate aim is to provide users with certain guarantees in terms of privacy and browsing experience, while preserving the online publishing's dominant business model.

Next, we summarize the major contributions of this work:

[1] These connections are often referred to as *third-party network requests*, while those established with the page's owner are called *first-party network requests*.

[2] Adblock Plus is Google Chrome's most popular plug-in the world with more than 50 million monthly active users, and an increase of 41 percent in the last year.

- We propose a theoretical model for the investigation of *behavioral targeting*, a widespread form of advertising that uses information gathered from users' Web-browsing behavior to serve them ads. The proposed model aims at providing transparency to this ad-serving process. First, by detecting such form of ad-targeting and thus quantifying the extent to which user-browsing interests are exploited. And secondly, by examining the uniqueness of the browsing profiles compiled by the entities that participate in said process.

  The strength of the proposed model lies in its more general and mathematically grounded approach to the problem of detecting such form of advertising. This is unlike previous work which relies on basic heuristics and extremely limiting assumptions, or which oversimplifies the ad-delivery process. The detection of behavioral advertising is, in this work, formulated as an optimization problem that reflects the uncertainty in determining the information available at ad platforms and trackers. The proposed model capitalizes on fundamental results from the fields of statistical estimation and robust optimization, the latter being a relatively new approach to optimization problems affected by uncertainty, but which has already proved useful in applications like signal processing, communication networks and portfolio optimization.

- In this same line of transparency and taking this model a step further, we propose a second detection system that sheds light on the uniqueness of the browsing profiles compiled by the entities that participate in the ad-delivery process. To this end, we adopt a quantifiable measure of user-profile uniqueness—the Kullback-Leibler (KL) divergence or *relative entropy* between the probability distribution of the user's Web-browsing interests and the population's distribution, a quantity that we justified and interpreted in [173, 179] by leveraging on the rationale behind entropy-maximization methods.

- We design a system architecture that implements the two aforementioned detection systems as main transparency factors, and enables smart ad blocking through the specification of user-configurable control policies. The system is designed to provide ad transparency and blocking services all in real-time, without the need of any external entity, and by relying on local Web-content categorization and open-source optimization libraries. The only exception is the computation of the profile uniqueness, which requires the involvement of an external server. A relevant aspect of our system is that it has been conceived to work under two distinct scenarios in terms of tracking, which allows users to configure the ad-transparency functionality according to their own perceptions in this respect. The proposed system architecture is developed in the form of a Web-browser extension for Google Chrome, and its beta version is available under request.

- We conduct an experimental analysis from the user data collected by this extension. Such analysis allows us, first, to evaluate the proposed system in a real environment; and secondly, to investigate several aspects related to behavioral advertising. The conducted experiments constitute the first attempt to study behavioral targeting from real user browsing profiles.

The remainder of this work is organized as follows. Sec. 5.2 provides the necessary background in online advertising. Then, Sec. 5.3 presents the theoretical model for the detection of interest-based ads and profile uniqueness. Sec. 5.4 describes the main components of a system architecture that aims at providing ad transparency and advanced ad-blocking functionalities. Sec. 5.5 analyzes the data collected by the proposed tool in an experiment with 40 participants. Sec. 5.6 reviews the state of the art relevant to this

work. Conclusions are drawn in Sec. 5.7. Finally, sections C.2 and C.1 of the appendix C show, respectively, the linear-program formulation of the interest-based ad detector, and the feasibility of this optimization problem.

5.2 Background in Online Advertising

This section examines the online advertising ecosystem, providing the reader with the necessary depth to understand the technical contributions of this work. First, Sec. 5.2.1 gives an overview of the main actors of this ecosystem. Afterwards, Sec. 5.2.2 describes how ads are served on the Web, and then, Sec. 5.2.3 provides a standard classification of the targeting objectives commonly available to advertisers. Finally, Sec. 5.2.4 presents one of the technologies enabling this ad-serving process. For a detailed, complete explanation on the subject, the reader is referred to [189].

5.2.1 Key Actors

The online advertising industry is composed by a considerable number of entities with very specific and complementary roles, whose ultimate aim is to display ads on Web sites. Publishers, advertisers, ad platforms, ad agencies, aggregators and optimizers are some of the parties involved in the ad-delivery process [206]. Despite the enormous complexity[3] and constant evolution of the advertising ecosystem, the process whereby ads are presented on Web sites is usually characterized or modeled in terms of publishers, advertisers and ad platforms [196, 147, 203, 68, 199]. Next, we provide a description of these three key actors:

- A *publisher* is an entity that owns a Web page (or a Web site) and that, in exchange of some economic compensation, is willing to place ads of other parties in some spaces of its page (or site). An example of publisher is The New York Times' Web site.

- An *advertiser* is an entity that wants to display ads on one of the spaces offered by a publisher, and is disposed to pay for it. Advertisers typically engage the services of one or several *ad platforms* (described below), which are the ones responsible for displaying their ads on the publishers' sites. As we shall explain later in Sec.5.2.2, there exist two ad-platform models, allowing users to have two different roles. In the traditional albeit prevailing approach, advertisers indicate the targeting objective/s most suitable for their campaigns, that is, to which users they want their ads to be shown. For example, an advertiser may want the ad platform to serve its ads to an audience interested in politics or to people living in France. Advertisers must also specify the amount of money they are willing to pay each time their ads are displayed, and each time users click on them[4]. On the contrary, in the recently established model of *real-time bidding* (RTB), ad platforms allow advertisers to bid for each ad-impression at the time the user's browser loads a page. This model enables advertisers to make their own decisions rather than relying on an intermediary to make decisions for them [189].

- An *advertising platform* or *ad platform* is a group of entities that connects advertisers to publishers, i.e., it receives ads from advertisers and places them on the spaces available at publishers. To this end, ad platforms track and profile users with the aim of targeting ads to their interests, location and other personal data. As we

[3] The intricacy of the advertising ecosystem is often illustrated in conferences and related venues with the diagram available at [132].

[4] In the terminology of online advertising, these quantities are referred to as the cost-per-impression (CPI) and the cost-per-click (CPC), respectively.

shall describe in greater detail in the next subsection, traditional ad platforms carry out this targeting on their own, in accordance with the campaign requirements and objectives specified by advertisers. RTB-based ad platforms, on the other hand, share certain user-tracking data with advertisers, which then take charge of selecting who suits them by deciding which user to bid for. Some examples of ad platforms include DoubleClick, Gemini and Bing Ads, owned respectively by Google, Yahoo! and Microsoft.

5.2.2 Ad-Serving Process

Without loss of rigor, throughout this work we shall assume an online advertising model composed mainly of the three entities set forth in the previous subsection. In this simplified albeit comprehensive terms, the ad-delivery process begins with publishers embedding in their sites a link to the ad platform/s they want to work with. The upshot is as follows: when a user retrieves one of those Web sites and loads it, their browser is immediately directed to all the embedded links. Then, through the use of third-party cookies, Web fingerprinting or other tracking technologies, the ad platform is able to track the user's visit to this and any other site partnering with it.

As one might guess, the ability of tracking users across the Web is of paramount importance for ad platforms: it enables them to learn the Web page being visited and hence its content; the user's location through their IP address; and, more importantly, their Web-browsing interests. Afterwards, all these invaluable data about the user is what allow ad platforms to serve *targeted* ads.

To carry out this task, the vast majority of ad platforms rely on proprietary *targeting algorithms* [189]. The aforementioned user data and the objectives and budgets of all advertisers for displaying their ads are the inputs of these algorithms, which are responsible for selecting which ad will be shown in a particular ad space. Evidently, their primary aim is to maximize ad-platforms' revenues whilst satisfying advertisers' demand.

As anticipated in Sec. 5.2.1, a new class of ad platforms has recently emerged that delegates this targeting process to external third parties, which then compete in real-time auctions for the impression of their ads. Ad platforms relying on this scheme usually share information about the user with these parties so that they can decide whether to bid or not for an ad-impression. Typically, the entities participating in these auctions are big advertising agencies representing small and medium advertisers[5], and traditional ad platforms wishing to sell the remnant inventory. This ad-serving scheme is called RTB and its major advantage, compared to the traditional ad platforms, is to enable advertisers (or others acting on their behalf) to buy individual impressions without having to rely on the ad platform's targeting decision. In other words, advertisers can decide whether a particular user is the right person to whom to present their ads.

Finally, regardless of the type of ad platform involved (i.e., RTB-based or not), the ad-serving process ends up by displaying the selected ad in the user's Web browser, a last step that may entail a content-delivery network.

Last but not least, we would like to stress that the advertising model described here —and considered in this work— corresponds to *indirect-sale advertising*, also called network-based or third-party advertising. This is in contrast to the *direct-sale advertisement* model, where publishers and advertisers negotiate directly, without the mediation of ad platforms. In this latter case, we mostly find popular Web sites selling ad space directly to large advertisers. The ads served this way are essentially untargeted, and are often displayed in Web sites where the products and services advertised are related to their

[5] A special class of these agencies are the *demand-side platforms* (DSPs), which are systems that automate the purchasing of online advertising on behalf of advertisers.

contents. This is mainly because the capability of a publisher to track and profile users is limited just to its own Web site and maybe a few partners. For example, the New York Times' Web site may track users also across the International Herald Tribune, owned by its media group. Such a tracking capability, however, is ridiculous when we compare it with the 2 million sites reachable by Google's ad platforms [72].

5.2.3 User-Targeting Objectives

The ads delivered through indirect-sale advertising allow advertisers to target different aspects of a Web user. The most popular *targeting objectives* include serving ads tailored to the Web page they are currently visiting, their geographic location, and their Web-browsing interests. Depending on the objective chosen by an advertiser, ads are classified accordingly as *contextual*, *location-based*, *interest-based* and *untargeted* ads. Occasionally, we shall refer to these four type of ads as *ad classes*. Next, we briefly elaborate on each them.

- *Contextual ads.* Advertisers can reach their audience through contextual and semantic advertising, by directing ads related to the content of the Web site where they are to be displayed. An example of such targeting strategy would be a health-insurance company wishing to show their ads in Web sites whose content is classified as "health & fitness".

- *Location-based ads.* They are generated based on the user's location, for example, given by the GPS of their smartphone or tablet, and also according to the Wi-Fi access points and IP address of the user's machine or device. Geographically-targeted ads enable advertisers to launch campaigns targeting users within a certain geographical location. An example would be the advertisement of a local music event to users reporting nearby locations.

- *Interest-based* or *profile-based ads.* Advertisers can also target users based on their Web-browsing interests. Usually, such interests are inferred from the pages tracked by ad platforms and other tracking companies that may share this information with the former. The sequence of Web sites browsed by a user and effectively tracked by an ad-platform or tracker is referred to as the user's *clickstream*. In current practice, this is the information leveraged by the online advertising industry to construct a user's interest profile [4, 197, 35, 147, 203, 68, 172, 189].

- *Generic ads.* Advertisers can also specify ad placements or sections of publisher's Web sites (among those partnering with the ad platform) where their ads will be displayed. Ads served through placement targeting are not necessarily in line with the Web site's content, but may simply respond to some match between the interests of the visiting users and the products advertised. Because these ads do not rely on any user data, we shall also refer to them as *generic ads*.

An important aspect of the ad-classes described above is that the former three are not mutually exclusive. In other words, except for placement ads —which are considered to be untargeted—, ads can be simultaneously directed based on content, location and interests. Accordingly, when we refer to interest-based ads, we shall mean that they are targeted *at least* to browsing-interests data. We shall refer to content- and location-based ads in an analogous manner.

In the terminology of online advertising, directing interest-based ads is often called *behavioral targeting*. Another quite popular ad-targeting strategy is *retargeting*, which helps advertisers reach users who previously visited their Web sites. For example, after

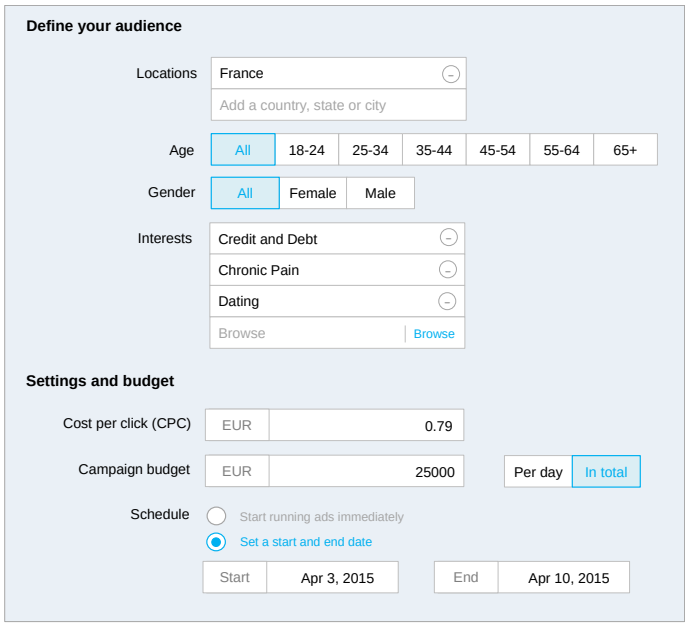


Figure 5.1: Gemini, Yahoo!'s ad platform, offers advertisers the possibility to target ads based on a number of parameters, including the user's browsing interests, which are chosen from a predefined set of 281 bottom-level categories. The categories selected in this example merely show the sensitive, personal information involved in these transactions, and thus do not reflect a real marketing campaign.

having browsed Apple's Web site, a user could be shown ads about a new iPhone release when visiting other sites, in an attempt to bring them back.

We conclude this subsection by giving a real-world example of how advertisers can target their ads. Fig. 5.1 shows the configuration panel available at Yahoo!'s ad platform, whereby advertisers can define their target audiences based on location, age, gender, interests[6] and context (not shown in this figure). For each campaign, the advertiser must configure all these variables appropriately, evidently with a constraint on the advertising budget.

5.2.4 Cookie Matching and Real-Time Bidding

This last subsection explains in greater detail some key operational aspects of RTB, an ad-serving scheme that accounts for 20% of digital ad sales [189] but that is expected to be the dominant advertising paradigm in the next years [41].

In Sec. 5.2.2, we mentioned that RTB-based ad platforms share user information with certain entities, which then may bid for the impression of their ads. The auction participants typically include agencies representing advertisers, DSPs and traditional ad platforms. To facilitate the sharing of information with these bidders, RTB relies on a *cookie-matching* protocol.

Generally speaking, cookie matching is a process by which two different domains link the user IDs that they have assigned to a same user and that they store in their respective cookies. Typically, the process is conducted as follows. When a user visits the former domain, this domain redirects their browser to the latter domain, including its user ID as a parameter in the URL. Then, upon receiving the request, the latter domain links this ID with its own ID for this user [104].

[6] Others platforms like Google's allow advertisers to specify further constraints such as the time of the day ads will be shown, their frequency of appearance to a same user and specific ad-placements.

Cookie matching finds its most common application in RTB, where it allows the ad platform and the bidder to match their cookies for a particular user [26]. Usually, the protocol is executed only if the bidder wins an auction and delivers its ad to this particular user. The matching permits the bidder to look up the user (if present) in its own database. Also, if subsequent ad-auctions are hold for this user, the bidder will learn that the user information provided in those auctions refer to this same matched user. We must emphasize that this is under the assumption that this bidder is among the recipients of the bid requests sent by the ad platform.

Having described the technology underlying RTB, next we briefly examine the overall functioning of Google's scheme, probably the most representative. The following, however, is also valid for other RTB-based ad platforms, although with slight variations irrelevant to this work.

When a user visits a Web site with an ad space served through RTB, an HTTP request is submitted to the ad platform, which subsequently sends *bid requests* to potential participants. We note that the number and type of participants involved may vary on a per-auction basis, at the ad platform's discretion. Within the bid request, the ad platform generally includes the following data: the URL of the page being visited by the user; the topic category of the page; the user's IP address or parts of it; and other information related to their Web browser [27]. Accompanying this information, Google's ad platform incorporates a bidder-specific user ID, which implies that different bidders are given different IDs for a same user. Other RTB-based ad platforms, alternatively, include their own user's cookies.

Upon receiving the bid request, the bidder may identify the user within its own database through the cookie or identifier. This is provided that the cookie-matching protocol has been executed previously for this user. Thanks to such cookie or identifier, the bidder can track them across those Web pages in which it is invited to bid. From those tracked pages, the bidder can therefore build a profile[7], maybe complementing tracking and other personal data it may have about the user.

The bid price is then set on the basis of the bidder's targeting objectives, that is, whether it aims to target users visiting certain site categories, browsing from a given location, and/or having some specific profile. To evaluate if the ad-impression meets such objectives, the bidder relies on the aforementioned profile and the information included in the bid request. If interested, the bidder submits a price to the ad platform, which finally, in a last step, allows the winning bidder to deliver the ad to the user. It is worth stressing that all this process of gathering user data, ad bidding and delivering is conducted in just tens of milliseconds.

5.3 Detection of Profile-based Ad-Serving and Profile Uniqueness

As described in the background section, ad platforms, tracking companies and also advertisers gather information about users (e.g., the visited pages and their location) while they browse the Web. Later, these and other data are leveraged to present ads targeted to the content of the pages browsed, their current geographic location and/or their interests. We also mentioned that ad platforms may as well deliver placement ads, which are considered generic or untargeted ads.

This section investigates a mathematical model that aims at quantifying to what extent the information gathered about a user's *browsing interests* is exploited afterwards by the

[7]DoubleClick's guideline specifies that, unless a bidder wins a given impression, it must not use the data for that impression to profile users [47]. Nevertheless, because no active mechanism is enabled to enforce this, nothing prevents a bidder from misusing such user data.

online advertising industry to serve them ads. The proposed model focuses on the detection of interest-based ads since they are the result, and probably the cause, of tracking and profiling users' browsing habits throughout the Internet, often without their knowledge [164] and consent[8]. It is important to remark that the conducted analysis is restricted to network-based advertisement, as the capability of publishers to track and profile users is, in general, limited to their sites.

In addition to determining if the displayed ads may have been targeted to a browsing profile, this section addresses another inescapable question related to profile targeting: *how unique* are we seen through the eyes of the companies displaying ads to us? As we shall elaborate on in Sec. 5.3.3, the risk of profiling as well as the uniqueness of the profiles built by these companies is closely linked to the risk of reidentification.

In the coming sections, we shall provide the conceptual basis and fundamental operational structure of two detectors that aim at (1) identifying profile-based ads from their interest categories; and (2) shedding light on the uniqueness of the profiles compiled by the entities that participate in the ad-delivery process. In doing so, we make a preliminary step towards studying the commercial relevance of our browsing history and quantifying its actual impact on user privacy. Later in Sec. 5.4 we shall present *MyAdChoices*, a Web-browser extension that capitalizes on these two detectors to bring transparency into said process and to enable selective and smart ad-blocking.

5.3.1 Statistical and Information-Theoretic Preliminaries

This section establishes notational aspects and recalls a key information-theoretic concept assumed to be known in the remainder of this chapter.

The measurable space in which a *random variable* (r.v.) takes on values will be called an *alphabet*. Without loss of generality, we shall always assume that the alphabet is discrete. We shall follow the convention of using uppercase letters for r.v.'s, and lowercase letters for particular values they take on. The probability mass function (PMF) p of an r.v. X is a function that maps the values taken by X to their probabilities. Conceptually, a PMF is a *relative histogram* across the possible values determined by its alphabet.

Throughout this work, PMFs will be subindexed by their corresponding r.v.'s in case of ambiguity risk. Accordingly, both $p(x)$ and $p_X(x)$ denote the value of the function p_X at x. Occasionally, we shall refer to the function p by its value $p(x)$. We use the notations $p_{X|Y}$ and $p(x|y)$ equivalently.

We adopt the same notation for information-theoretic quantities used in [91]. Concordantly, the symbol D will denote relative entropy or KL divergence. We briefly recall this concept for the reader not intimately familiar with information theory. All logarithms are taken to base 2.

Given two probability distributions $p(x)$ and $q(x)$ over the same alphabet, the *KL divergence* $\mathrm{D}(p \,\|\, q)$ is defined as

$$\mathrm{D}(p \,\|\, q) = \sum_x p(x) \log \frac{p(x)}{q(x)}.$$

The KL divergence is often referred to as *relative entropy*, as it may be regarded as a generalization of the Shannon's entropy of a distribution, relative to another.

Although the KL divergence is not a distance in the mathematical sense of the term, because it is neither symmetric nor satisfies the triangle inequality, it does provide a measure of discrepancy between distributions, in the sense that $\mathrm{D}(p \,\|\, q) \geqslant 0$, with equality if, and only if, $p = q$.

[8] Consistently with the recommendations of the US Federal Trade Commission, the advertising industry has started to offer an opt-out scheme for behavioral advertising [7].

5.3.2 Detection of Profile-based Ads

One of the key functionalities of our system is the detection of profile-based ads, that is, ads that are tailored to a user's browsing interests and, in addition but not necessarily, to their location and the Web-page currently visited. This section proposes a mathematical model for the identification of these ads, which leverages fundamental results from statistical estimation and robust optimization.

Ad-Serving Interest-Category Model

We model the ads delivered by an ad platform (RTB-based or not) to a particular user as independent r.v.'s taking on values on a common finite alphabet of categories or topics, namely the set $\mathscr{X} = \{1, \ldots, n\}$ for some integer $n > 1$. We hasten to stress that our model encompasses the four classes of ads, or objectives, described in Sec. 5.2.3. The fact that each ad is associated with an interest category does not mean we are considering just interest-based ads. For example, a content-based ad displayed on the Web site `www.webmd.com` will be necessarily classified into an interest category related to health. Location-based and placement ads can evidently be mapped to any of the n categories assumed in this work.

As commented in Sec. 5.2.2, the ad-serving process takes into account a wide range of variables when displaying an ad to a user on a given ad space. These variables include tracking and profiling data about the user in question, the publisher being visited, the advertisers and their corresponding campaigns, and, depending on the ad-platform type, the bids of the ad-auction participants or the criteria of the ad platform itself to maximize its revenue.

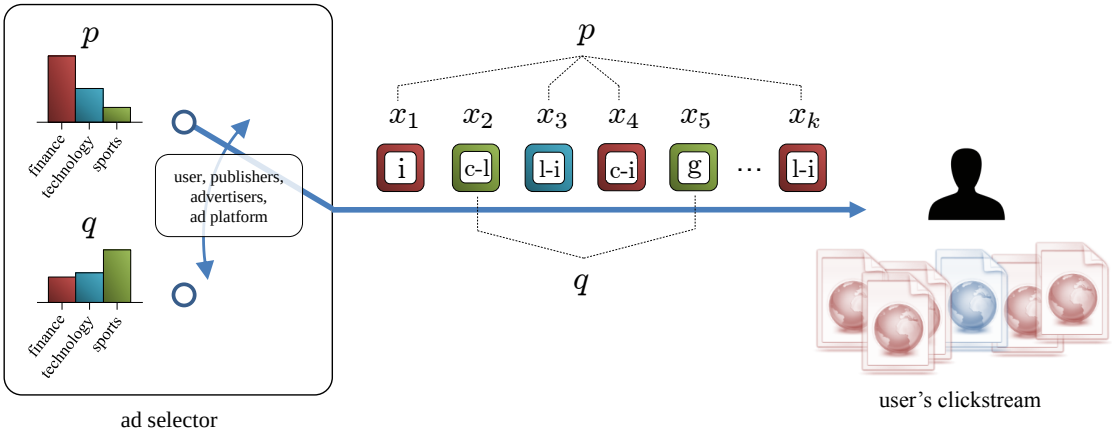


Figure 5.2: An ad selector (e.g., a traditional ad platform) displays k ads on the user's browser when navigating the Web. The interest categories of the delivered ads are modeled as a sequence of independent r.v.'s taking on values on $n = 3$ categories. The observed categories, i.e., $(x_i)_{i=1}^{k}$, can be seen as generated by a source that commutes between the PMFs p and q. The switching between interest-based ads (i.e., "i", "c-i", "l-i" and "c-l-i") on the one hand, and non-interest-based ads (i.e., "c", "l", "g" and "c-l") on the other, is determined by a number of parameters related to the user, publishers, advertisers and ad platform.

In our mathematical model, we characterize the ad-serving process conducted by an ad platform as a black box, whose inputs are the variables mentioned above, and whose outputs are the selected ads. We explained in the background section that traditional ad platforms are the ones selecting the ad to be displayed, while in RTB-based advertising the choice is made by the winning bidder, being an advertising agency or a traditional ad platform. For the sake of conciseness and to avoid specifying the ad-platform model in

each case, we shall henceforth use the term *ad selector* to refer generically to the particular entity imposing the selection of an ad.

For each user and for each ad space, the outputted ads can be classified as content-, location-, and interested-based and generic, according to the corresponding advertisers' targeting objectives. We note that, from these four classes of ads, we may only have eight possible combinations of those classes. Denoting each of the ad-classes by its first letter, the set of all such combinations is

$$\mathscr{G} = \{\text{c}, \text{l}, \text{i}, \text{g}, \text{c-l}, \text{c-i}, \text{l-i}, \text{c-l-i}\},$$

where the element "c-l" represents an ad that has been targeted based on content and location. In other words, $\mathscr{G}$ includes all the combinations of targeting objectives an advertiser may choose.

We mentioned in Sec. 5.2.2 that user profiles are essentially built from clickstreams, i.e., from the Web pages tracked. For $k >> 1$, let $(X_i)_{i=1}^{k}$ be the sequence of ads that an ad selector (e.g., a traditional ad platform) delivers to a particular user during several browsing sessions. Our characterization of this ad-delivery process stems from the intuitive observation that, if we were able to rule out all but the interest-based ads of such sequence, the empirical distribution [91] of the interest categories observed would naturally resemble, to a large extent, the user's browsing interests, or equivalently, their clickstream.

According to this observation and without loss of generality, we model the sequence of outgoing ads, classified into interest categories, as the output of an ad-source that *alternates* between two probability distributions, namely

- an interest-category distribution p that reflects the knowledge the ad selector has about the *user's interests*;
- and another interest-category distribution q that represents the complement of the former distribution and thus corresponds to (the interest categories of) those ads classified as non-interest-based, that is, *contextual*, *location-based* and *generic*.

Naturally, the model described above captures only one aspect of the ad-serving process: it reflects the selection of the ads interest-categories within the set $\mathscr{X}$, a step that we model through the distributions p and q when the ad-class is respectively interest-based and non-interest-based. The proposed model is supported by the reasonable assumption that the accumulated interest categories of the interest-based ads will very likely approximate to the user's interests, or more precisely, the clickstream possessed by the ad selector.

Our model does not, therefore, capture other aspects of the ad-serving process like how a particular ad-class combination is chosen from $\mathscr{G}$. With it, however, we reflect the simple fact that the interest categories of the outgoing ads may be distributed according to either partial (or complete) user browsing data, or any other information which does not include those browsing data. This simplified ad-serving model based on interest categories will allow us in the next subsection to estimate the ad-class chosen by the ad selector, or more accurately, whether the delivered ads are classified as interest-based or not. Fig. 5.2 illustrates how we model this aspect of the ad-serving process.

Binary Hypothesis Testing

Assuming such model on the ad-platform's side, on the user's side we aim to determine if an ad, previously classified into an interest category, has been shown to the user based on their past Web-browsing interests or not. Formally, we may consider this as a binary *hypothesis testing* problem [91] between two hypothesis, namely whether the data (i.e., the category of the displayed ad) has been drawn according to the distribution p or q. Next, we elaborate on these two distributions. Further details about the practical estimation of both PMFs are set forth in Sec. 5.4.

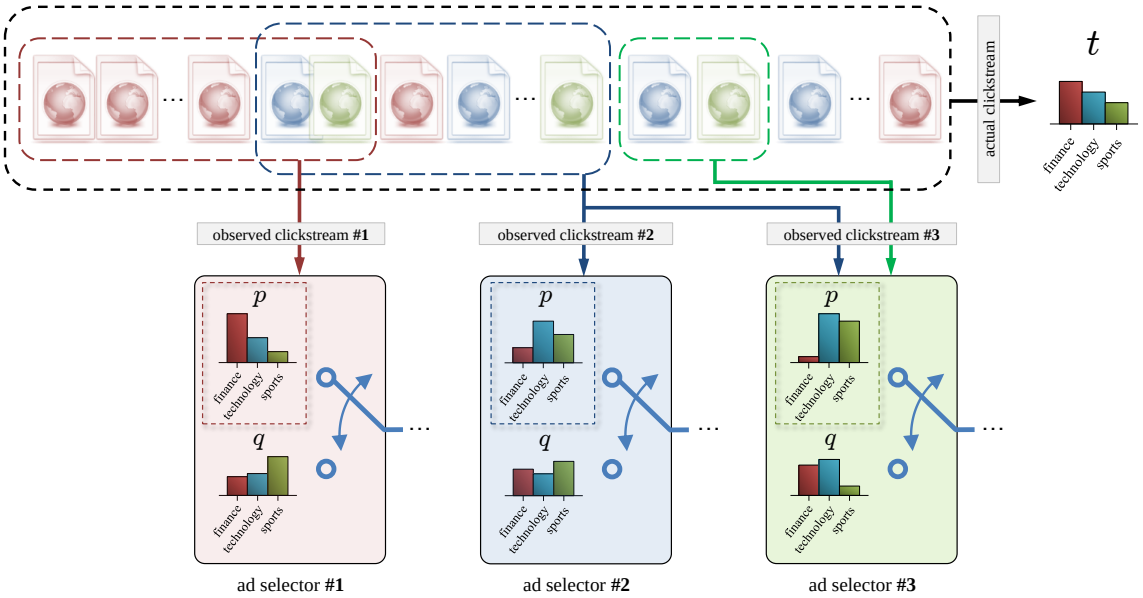


Figure 5.3: We show how three ad selectors track a user through different Web sites. The ad selectors 1 and 2 could represent two ad platforms overlapping their observed clickstreams. This would reflect a common situation for large ad platforms like Google AdSense and OpenX. The ad selector 3, on the other hand, could exemplify a small advertising company. Because of its limited ability to track users on its own, this latter ad selector might decide to acquire tracking data from the ad selector 2. Regardless of the data exchanged, however, none of the three ad selectors will be able to get the actual clickstream.

Recall that, for a particular user and ad space, the *ad selector* is the entity that ultimately decides which ad is shown to that user in that ad space. In the case of traditional ad platforms, the ad selector is the ad platform itself. In RTB, on the contrary, the ad selector is the bidder that wins the auction for displaying its ad, being an agency representing advertisers, a DSPs or a traditional ad platform.

As described in Sec. 5.3.2, the PMF p represents the knowledge that such ad selector has about the user's browsing interests. Henceforth, we shall refer to this distribution as the user's *interest profile*, bearing in mind that it is specific to the ad selector in question.

In practice, these profiles are typically built from the tracked Web sites or *observed clickstream* [4, 197, 35, 147, 203, 68, 172, 189]. The clickstream available to an ad selector, however, need not necessarily be the result of a direct tracking on the user. For example, ad platforms may track users on their own through their cookies; and not satisfied with that, they may also wish to build upon tracking data from other ad platforms or trackers. For the time being, we shall not specify how, in our model, the ad selector profiles a user from their clickstream. We shall only assume that profiles are represented as PMFs, as many works in the literature essentially do [197, 176, 147, 203, 68].

Clearly, depending on the ability of the ad selector to track users throughout the Web (on its own or not), the profile p will resemble, to a greater or lesser extent, their actual interests. We denote by t the interest profile resulting from the *actual clickstream*, that is, all the Web sites visited by a user. We shall occasionally refer to p and t as the *observed* and *actual profiles*, respectively. Fig. 5.3 extends the ad-targeting model depicted in Fig. 5.2, to reflect the fact that p is constructed from the observed clickstream and thus may not capture the user's actual interest profile t.

The distinction between these two profiles will also be employed later in Sec. 5.4 to reflect two possible scenarios regarding tracking and sharing of clickstream data: on the one hand, a paranoid scenario where users are tracked on every page they visit and such

tracking data is exchanged among all entities serving ads. And on the other hand, a baseline scenario where p is fundamentally built from the clickstream an ad selector may get on its own, through cookies or other tracking technologies, without relying on tracking data from other sources.

In order to conduct our hypothesis testing, we shall also need to estimate the distribution q. To this end, we consider an environment where no tracking is performed, similarly to when users enable the Web-browser's private mode. Recall that this PMF is the interest-category distribution of those ads which are not profile-based, that is, those classified as "c", "l", "g" and "c-l". Because, except for ad-placement, these ads will depend on the user's location and the pages visited during this free-tracking session, q will be specific to each particular user. To estimate this distribution on the user side, we shall capture the category of all ads received, under the reasonable assumption that, when users browse in private mode, no browsing-interest data are leveraged to target the ads. In Sec. 5.4.2, we shall describe more specifically how this PMF will be estimated by our detector.

Short-Term and Long-Term Interest Profiles

In previous sections, we pointed out that user interest profiles are mainly built from the categorization of the visited Web sites. We also commented that profiles are modeled essentially as PMFs, that is, as histograms of relative frequencies of those visited sites across a set of interest categories. In this subsection, we briefly examine a crucial aspect of such user modeling, namely, we explore the importance that ad selectors may place on recent interests compared to those accumulated over a long time period.

From the perspective of profile-based targeting, the need to weight clickstreams is evident. A short recent history may be enough to direct products which do not require much thought, like buying a movie at Google Play. But other kind of transactions such as enrolling for an online university degree may need a longer browsing history to ensure a certain probability of conversion[9] [172].

Depending on the time window chosen, user profiles can be classified as *short-term* and *long-term* profiles. The former represent the user's current and immediate interests, whereas the latter capture interests which are not subject to frequent changes [114]. In general, different interest-based marketing systems may contemplate different time windows for building profiles. Many commercial systems opt for relatively long-term profiles, while others capitalize on short, recent clickstreams. Some recent studies do not seem to agree on that, either. For example, [172] provides evidence that long browsing histories may lead to better targeting of users, while others show the opposite [203].

As we shall see in Sec. 5.3.2, our detection system will capture the uncertainty associated with the time window used by an ad selector. Since in practice it is impossible to ascertain this parameter, we shall consider uncertainty classes of user profiles. These classes will enable us to characterize the distinct options an ad selector might have chosen to create a profile, and will lead us to the design of an optimal robust detector.

Optimal Detection of Interest-based Ads under Uncertainty

In this section, we formulate the problem of designing an interest-based ad detector as a robust minimax optimization problem. To this end, we essentially follow the methodology developed by [82, 145].

Let X be an r.v. modeling the category an ad belongs to. Denote by H the r.v. representing the two possible hypothesis about the distribution of the observed category X. Let $H = 1$ indicate that the ad is profile-based (first hypothesis), and $H = 2$ it is not

[9] In online marketing terminology, conversion usually means the act of converting Web site visitors into paying customers.

profile-based (second hypothesis). Said otherwise, X conditioned on H has PMF p when $H = 1$ and q when $H = 2$. For the sake of compactness, we denote by $P \in \mathbb{R}^{n\times 2}$ the matrix that has p and q as columns.

A *randomized estimator* or *detector* $\hat{H}$ of H is a probabilistic decision rule determined by the conditional probability of $\hat{H}$ given X, that is, $p_{\hat{H}|X}$. The interpretation of such estimator is as follows: if X is observed to have value j, the detector concludes $H = 1$ with probability $p_{\hat{H}|X}(1|j)$, and $H = 2$ with the complement of that probability.

A randomized detector also admits an interpretation in matrix terms, in particular as an $\mathbb{R}^{2\times n}$ matrix, where the j-th column corresponds to the probability distribution of $\hat{H}$ when we receive an ad belonging to the interest category j. Throughout this section, we shall conveniently use this matrix notation for estimators, and denote by D the matrix defining them.

The performance of a decision rule is usually characterized in terms of its detection and error probabilities. We may capture this performance compactly by means of the matrix $M = DP$, whose element M_{ij} gives us the probability of deciding $\hat{H} = i$ when in fact $H = j$, that is, $p_{\hat{H}|H}(i|j)$. The diagonal elements of this 2×2 matrix are the probabilities of correct guess. The error probabilities are represented by the off-diagonal elements M_{21} and M_{12}, which yield the probabilities of a false negative and a false positive, respectively. In our context, the former is the probability of concluding that the ad is not profile-based when actually it is; and the latter is the probability of deciding the ad is interest-based when it is not.

Our aim is to design the matrix D that defines the interest-based ads detector, so that certain performance criteria are satisfied. Among other requirements, we might be interested in minimizing (maximizing) one of the error (detection) probabilities, with a constraint on the complement of the objective probability. Also, we could consider minimizing both error probabilities or a convex combination of them, if some prior information about p_H was available.

Robust Estimation Regardless of the criteria chosen, the problem of this design is that it requires the complete knowledge of the probability distributions defined by P. As explained in the previous section, we may compute a reliable estimate of q locally (i.e., on the user side), but we cannot know how ad selectors construct the profile p from their observed clickstream. Some ad selectors may wish to target users based on their short-term interests, some may rely on longer and relatively stable profiles to this end, and others may opt for both kind of models. In any case, the time window/s employed by an ad selector is what determines the profile/s that will be used for ad targeting. Because this information is unknown, having a precise specification of the distribution p, or estimating it reliably, is therefore infeasible.

The problem of estimating a distribution under uncertainty has also been encountered in other fields and applications such as signal processing [210], portfolio optimization [163] and communications networks [204]. In all these cases, the probability distributions are frequently specified to belong to sets of distributions, typically called *uncertainty classes*. In our case, the uncertainty class of p is given by the minimum and the maximum lengths of the time windows an ad selector may define to model short-term and long-term interests. In practice, the maximum length might correspond to the entire clickstream, whereas a minimum reasonable time window for short-term profiles might be one day [172, 68].

For $i = 1, \ldots, n$, we denote by $p_i^{\max}$ the maximum interest value p_i estimated by the ad selector, over *all* possible time windows ranging from one day to the whole observed clickstream[10]. We define $p_i^{\min}$ analogously, and intuitively model the uncertainty about the distribution p as intervals between these upper and lower bounds. More specifically,

[10]In Sec. 5.4.2, we shall see that a maximum time window of 1.5 months may be sufficient.

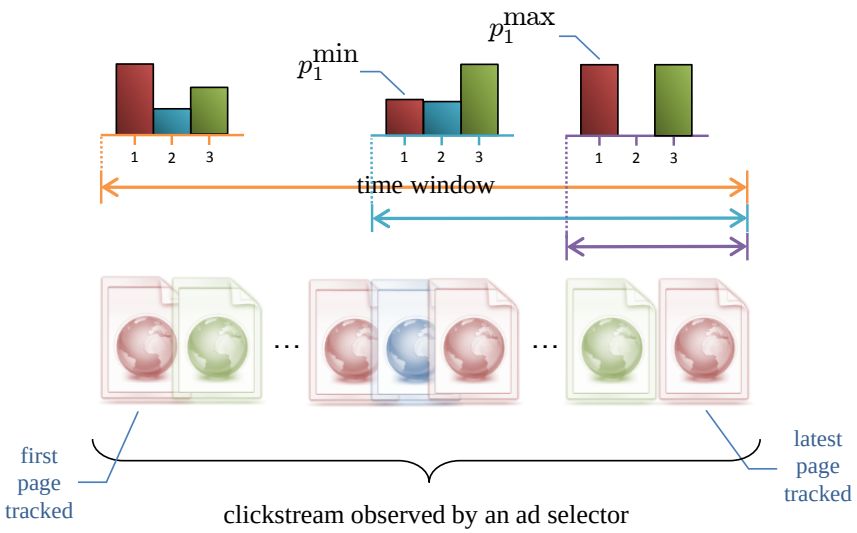


Figure 5.4: Ad selectors may create interest profiles based on the Web pages tracked. Our detector captures all possible options an ad selector may consider to compute those profiles from the tracked pages. All these options are directly related to the time window/s chosen, or equivalently, the number of pages taken from the observed clickstream. We model these possible choices as intervals between minimum and maximum interest values per category.

we define the set of possible interest profiles as

$$\mathcal{P} = \{p : p^{\min} \preceq p \preceq p^{\max}, \mathbf{1}^{\mathrm{T}} p = 1, p \succeq 0\}, \tag{5.1}$$

where the symbol "$\preceq$" indicates componentwise inequality, and the last inequality and the equality reflect the fact that p must be a PMF.

At a conceptual level, the polyhedron $\mathcal{P}$ captures all the possible profiles that an ad selector may have built by adding incremental observations of one Web site to the interests model. By computing the maximum and minimum observed interests over all these incremental models, and by defining intervals of interest values between these two extremes, we obtain an uncertainty class that reflects *any* possible decision made by the ad selector regarding the time window. We would also like to stress that the uncertainty class $\mathcal{P}$ likewise includes the possibility that an ad selector may be using more than one profile —with different time windows— for a same user. Fig. 5.4 illustrates the uncertainty around the selected time window/s.

One possible way to devise an estimator when a probability distribution is specified to belong to an uncertainty class is to contemplate the *worst-case* performance over this class. The resulting decision rule is then said to be *robust* to the uncertainties in the probability distribution [134]. Following the notation of [82], we define the worst-case performance matrix M^w associated with a robust detector as

$$M_{ij}^w = \sup_{p \in \mathcal{P}} M_{ij},$$

for $i, j = 1, 2$, with $i \neq j$, and

$$M_{ii}^w = \inf_{p \in \mathcal{P}} M_{ii},$$

for $i = 1, 2$. In general terms, the off-diagonal elements of this matrix give us the largest probability of errors over all $p \in \mathcal{P}$. The diagonal entries, on the other hand, yield the smallest possible detection probabilities. Based on the latter probabilities, we may define the *worst-case error probability* as $P_i^w = 1 - M_{ii}^w$, which represents the largest probability of error over the uncertainty class when $H = i$. Clearly, we note that $M_{12}^w = M_{12}$ and $M_{22}^w = M_{22}$, as in our case the uncertainly is just in p.

Minimax Design In this subsection, we specify the design of a robust interest-based ad detector, and formulate the hypothesis test problem between H_1 and H_2 as a linear program (LP).

Based on the error and detection probabilities shown in the previous subsection, various designs can be developed. Some classical optimality criteria are the Bayes, Neyman-Pearson and minimax designs [145]. In this work, we consider a robust minimax approach that minimizes the worst-case error probability, over the two hypotheses. We adopt this approach because, in our attempt to detect interest-based ads, both error probabilities are equally important.

According to this design criterion, the proposed robust minimax detector is given by the matrix D that solves the optimization problem

$$\min \max_{i=1,2} P_i^w. \tag{5.2}$$

Let $\tilde{d}^{\mathrm{T}}$ be the first row of D, that is, the conditional probabilities $p_{\hat{H}|X}(1|j)$ for $j = 1, \ldots, n$. We show in the section C.2 of the appendix C that (5.2) is equivalent to the following optimization problem in the variables $\lambda, \mu, \tilde{d} \in \mathbb{R}^n$ and $\nu \in \mathbb{R}$:

$$\begin{aligned} \text{maximize} \quad & \zeta \\ \text{subject to} \quad & \mu^{\mathrm{T}} p^{\min} - \lambda^{\mathrm{T}} p^{\max} + \nu \geqslant \zeta, \\ & 1 - \tilde{d}^{\mathrm{T}} q \geqslant \zeta, \\ & \mu - \lambda + \nu \mathbf{1} \preceq \tilde{d}, \\ & \lambda \succeq 0, \mu \succeq 0, \\ & 0 \preceq \tilde{d} \preceq \mathbf{1}. \end{aligned} \tag{5.3}$$

The strength of recasting (5.2) as an LP lies in that it allows us to resort to extremely efficient and powerful methods to compute the optimal detector. This is of a great practical relevance as we aim to provide such interest-based detection functionality on the user side, as a stand-alone software operating in real-time, i.e., while the user browses the Web. Sec. 5.4 will give further details about the optimization library used for this computation. The feasibility of this optimization problem is shown in the section C.1 of the appendix C.

5.3.3 Detection of Profile Uniqueness

In the previous subsection, we provided the design of a robust interest-based detector whereby users may learn to what extent their browsing profiles are exploited to serve them ads. This subsection investigates another crucial aspect related to behavioral targeting, namely, if the profiles collected by the advertising companies might reveal *unique* browsing patterns.

The importance of this aspect lies in the potential risk of reidentification from unique, non-personally identifiable data, as illustrated, for example, by the *AOL* search data scandal [69][11]. In our context, the risk of profiling goes hand in hand with the risk of reidentification, especially when considered in the context of additional information obtainable from a user such as their location, accurate navigation timing and aspects related to the Web browser and operating system. When the profile is added also to the wealth of data shared across numerous information services, which a privacy attacker could observe and cross-reference, such attacker might eventually find out, even if in a statistical sense, the user's real identity.

Having motivated the risk of profile uniqueness, this subsection describes how to detect if the ads delivered to a user may have been generated as a result of a common browsing

[11] *AOL* user No. 4417749 found this out the hard way in 2006, when *AOL* released a text file intended for research purposes containing twenty million search keywords including hers. Reporters were able to narrow down the 62-year-old widow in Lilburn, Ga., by examining the content of her search queries [69].

pattern, or conversely, to a browsing history that deviates from a typical behavior. To this end, we first provide a brief justification of KL divergence as a measure of the uniqueness of a profile, or equivalently, its commonality. The rationale behind the use of divergence to capture this aspect of a profile is documented in greater detail in [173, 179]. Afterwards, we examine how to estimate this information-theoretic quantity.

Although we mentioned in Sec. 5.3.1 that the KL divergence is not a proper metric, its sense of discrepancy between distributions allows an intuitive justification as a measure of profile commonality. Particularly, whenever the profile observed by an ad selector diverges too much from the average profile of all tracked users, the ad selector will be able to ascertain whether the interests of the user in question are atypical, in contrast to the statistics of the general population.

A richer justification arises from Jaynes' celebrated *rationale on entropy maximization methods* [128, 127], which builds on the method of types [91, §11], a powerful technique in large deviation theory. Leveraging on this rationale, the relative entropy between an observed profile and the population's profile may be considered as a measure of the uniqueness of the former distribution within such population. The leading idea is that the method of types establishes an approximate monotonic relationship between the *likelihood* of a PMF in a stochastic system and its divergence with respect to a reference distribution, say the population's. Loosely speaking and in our context, the lower the divergence of a profile with respect to the average profile, the more likely it is, and the more users behave according to it. Under this interpretation, the KL divergence is therefore interpreted as an (inverse) indicator of the commonness of similar profiles in said population.

Having argued for the use of KL divergence as a measure of profile commonality, next we elaborate on the uncertainty to estimate this divergence value. Recall from Sec. 5.3.2 that ad selectors may construct profiles in multiple ways from the observed clickstream. Just as we did with the design of the interest-based ad estimator, we proceed by considering a worst-case uniqueness estimate on the space of possible profiles built by an ad selector. Denote by $\bar{p}$ the *population's interest profile*. Formally, for each user and ad selector, we define the *minimum uniqueness* over all such profiles as

$$u_{\min} = \inf_{p \in \mathscr{P}} \mathrm{D}(p \,\|\, \bar{p}), \tag{5.4}$$

which gives a measure of profile commonness that allows for the uncertainty inherent in the time window used by an ad selector.

The divergence-minimization problem above captures a worst-case scenario regarding profile commonality. In particular, it tells us how peculiar our interests might be, as seen by an ad selector. For any ad selector, the value $u_{\min}$ (on the interval $[0, \infty)$ bits) will clearly vary over time as the user browses the Web. From the point of view of comprehensiveness, however, the information conveyed each time by this *absolute* uniqueness value may not be informative enough to the user.

To help the user interpret a given $u_{\min}$ value, we consider making it *relative* to a population of users. In doing so, users can compare their profile uniqueness values with those of other users of our Web-browser extension, and thus gain a broader perspective of how they are profiled. Also, users may utilize this information to define consequent ad-blocking policies. Later in Sec. 5.4, we shall describe the exchange of information between users of our system and a central repository to estimate those relative profile-uniqueness values.

5.4 "MyAdChoices" — an Ad Transparency and Blocking Tool

This section describes *MyAdChoices*, a prototype system that aims to bring transparency into the ad-delivery process, so that users can make an informed and equitable decision regarding ad blocking. The proposed system provides two main functionalities. Enabled by the interest-based ad detector and the profile-uniqueness estimator designed in Sec. 5.3, the ad-transparency functionality allows users to understand what is happening behind the scenes with their Web-browsing data. The ad-blocking functionality, on the other hand, permits users to react accordingly, in a flexible and non-radical manner. This is unlike current ad-blocking technologies, which simply block or allow all ads. *MyAdChoices* does not only consider these two extremes, but the interesting and necessary continuum in between. With this latter functionality, users can indicate the type of ads they wish to receive or, said otherwise, those which they want to block. By combining both functionalities and thus providing transparency and fine-grained control over online advertising, the proposed system may help preserve the Internet's dominant economy model, currently threatened by the rise of simple, radical ad blockers.

This section is organized as follows. Sec. 5.4.1 first elaborates on the ad transparency and blocking functionalities provided by our system. Afterwards, Sec. 5.4.2 describes the components of a system architecture that implements these two functionalities.

5.4.1 Main Functionalities

Our system brings *transparency* to two central aspects of behavioral ad-serving. On the one hand, it allows users to know if the information gathered about their browsing interests may have been utilized by the advertising industry to target them ads. Specifically, our system lets the user know if the received ads may have been generated according to their browsing interests or, more accurately, to the profiles that ad selectors may have about them. On the other hand, it provides insight into the browsing profiles that ad selectors may have inferred from the pages tracked. In particular, *MyAdChoices* shows a worst-case, profile-uniqueness value for each ad selector, and the interest category of the ads received.

With regard to the *ad-blocking* service, our system contemplates the following user-configurable parameters:

- **Ad interest-category**. We offer users the possibility to filter ads by interest category. For example, a user could block ads belonging to certain sensitive categories like pornography and health.
- **Ad class**. This parameter enables users to block either the interest-based ads or the non-interest-based ads, for all ad interest-categories or for a subset of them.
- **Profile uniqueness**. Users may decide blocking the ads delivered by those ad selectors that may have compiled very unique, and thus potentially re-identifiable, profiles of their browsing habits.
- **Retargeting**. Last but not least, users can decide to block retargeted ads, that is, ads coming from advertisers that have been previously visited by the user (see Sec. 5.2.3).

Examples of Ad-Blocking Policies

This subsection provides a couple of simple but insightful ad-control policies that aim to illustrate the parameters described in the previous subsection. These examples are prefaced by a general definition of ad-filtering policy, inspired from the field of access control.

Definition 3 (Ad-blocking policy)**.** *A* policy *pol is a pair* $(AC, sign)$*, where* AC *is an* ad constraint*, and* $sign \in \{+, -\}$ *models an* action *to be taking when an ad meets that constraint. An ad constraint is represented by a triple* (I, i, u)*, where* $I \in \{0, 1\}$ *indicates if an ad is interest-based or not,* $i \in \mathscr{X}$ *is an interest category, and* $u_{\min}$ *denotes a requirement of minimum profile uniqueness.*

An ad constraint represents the set of ads belonging to an interest category i, which are classified as interested-based (or not), and which have been delivered according to a profile with minimum uniqueness given by $u_{\min}$. On the other hand, *sign* denotes if the ad must be blocked (-), or displayed on the user's browser (+).

Because the support for positive and negative policies may cause conflicts (i.e., we may have an ad constraint satisfying both positive and negative policies), a conflict-resolution mechanism must be enforced. The literature of access control provides several approaches to tackle such conflicts. A comprehensive survey on this topic is [110]. Here, for simplicity, we assume that negative policies prevail, since this approach provides stronger guarantees with regard to the risk of displaying unappropriate ads. Other conflict resolution policies, however, could also be readily integrated.

Two examples of policies are given next. In these examples, we refer to some of the interest categories used by the proposed system (see Sec. 5.4 for more details). For brevity, in this section we shall denote the relevant categories by its name. Also, for simplicity and clarity, in the examples we shall keep using the policy formal notation introduced in Definition 3. We note, however, that this notation, describing how policies are actually implemented in the system, must be made transparent in the front end both to improve usability and to help users specify policies reflecting as much as possible their preferences. As we shall explained in Sec. 5.4.2, several strategies will be devised for this purpose, e.g., the use of textual labels instead of numeric values.

Example 1 (Policies for allowing certain personalized ads)**.** *Alice had planned to visit New York City (NYC) for her holidays. Some days ago she bought her flight tickets and booked her hotel, all through the Internet. During the following days, she visited several Web sites in search of sightseeing tours and day trips. As she browsed the Web, the ads displayed in her browser became increasingly related to her upcoming trip. Alice is now fed up with ads on hotels in NYC, so she is considering installing AdBlock Plus to block them all. However, she appreciates the value and usefulness of behavioral targeting, and because she has not decided her itinerary yet, she still wants to receive personalized ads associated with the categories 1 ("Travel\| Trains") and 2 ("Travel\| Theme parks"). Consequently, Alice specifies the following policies:*

- $pol_1 = ((c_1, 1, \cdot), +)$,
- $pol_2 = ((c_2, 1, \cdot), +)$,

where the symbol "$\cdot$" means that the value of the parameter in question is not specified.

Example 2 (Policy for balancing personalization and privacy)**.** *Bob works in a dietetics and nutrition shop. As part of his work, he sometimes consults pages about health and fitness. Occasionally, and when nobody sees him, he spends some time checking Web sites related to his recently diagnosed fibromyalgia's disease. Some days ago he was shocked when a couple of ads on biological treatments for his disease popped up while he was browsing the Web. Since then Bob is very concerned that related ads may be displayed when his workmates look over his monitor. However, despite his worries, he does not wish to resort to the typical ad-blocking plug-ins, as such personalized-ads services also helps him keep abreast of the newest products and trends in his work. To strike a balance between privacy*

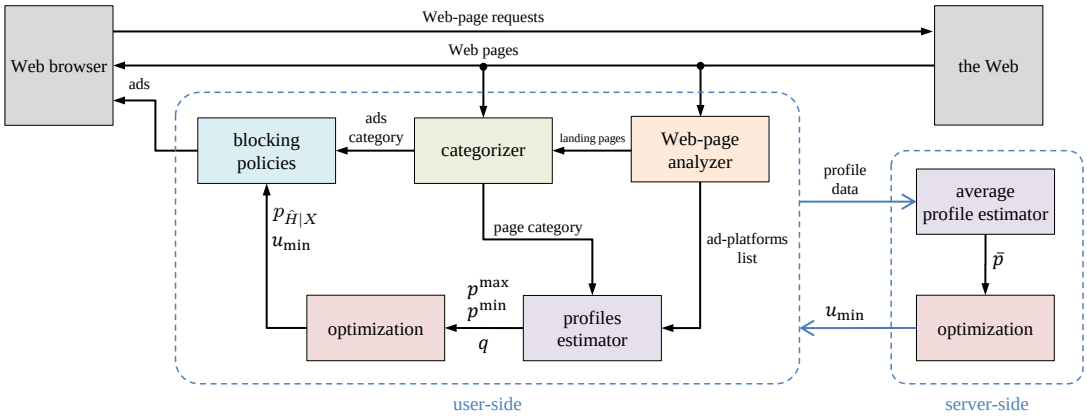


Figure 5.5: Internal components of the proposed architecture.

and personalization, Bob specifies a filter that blocks profile-based, health-related ads only when his browsing profile reflects relatively atypical interests. In particular, he defines the following policy:

- $pol_1 = ((c_3, 1, \pi_{u_{\min}} \geqslant 25\%), -)$,

where the category 3 corresponds to "health & fitness", and $\pi_{u_{\min}}$ denotes the percentile value of $u_{\min}$.

Lastly, we would like to emphasize the topicality and appropriateness of this latter example, with an extreme case in which a cancer patient reported numerous Facebook ads for funeral companies after having searched for his recently diagnosed disease [202].

5.4.2 System Architecture and Implementation Details

In this section, we describe the components of a system architecture that implements the two functionalities specified in Sec. 5.3. The proposed system has been developed as a Web-browser extension and is available for Google Chrome[12]. It is worth emphasizing that this extension not only provides transparency and ad-blocking services in real-time, but also operates as a stand-alone system, i.e., it performs all computations and operations locally, without the need of any infrastructure or external entity to this end. The only exception is the computation of the minimum profile-uniqueness value, which is not done on the user side, as it requires the average profile of the population $\bar{p}$. As we shall elaborate later on in Sec. 5.4.2, this particular service is provided only if the user accepts sharing profile data with the *MyAdChoices* server.

Assumptions

Before proceeding with the description of the system architecture, we examine the assumptions made in implementing the interest-based ad detector and the profile-uniqueness estimator designed in Secs. 5.3.2 and 5.3.3.

Our first assumption is related to the impossibility of finding out, with absolute certainty, the browsing information that ad selectors have about users. In Sec. 5.3.2 we called this information the observed clickstream, and defined it more precisely as the sequence of Web pages the ad selector knows that the user visited. By observing the third-party network requests, our browser extension is able to capture the pages ad platforms may track

[12]Currently, the tool is in beta version and can be downloaded at `https://myrealonlinechoices.inrialpes.fr` under request.

through HTTP cookies or other more sophisticated methods like Web-browser fingerprinting. Nevertheless, we cannot know if this is all the information available to them, i.e., if those pages account for their observed clickstreams or not —ad selectors and Web trackers may also exchange their tracking data, for example, through cookie matching, a practice that appears to be much more common than those direct tracking methods [165, 164, 58]. The fact that a cookie-matching protocol is executed between two entities does not imply, however, that they end up exchanging their tracking data. There is an obvious incentive to aggregate information and gain further insight into a user's browsing history, but since this exchange does not go through the user's browser, we cannot safely conclude that it is made.

In the case of RTB, the bid requests sent by an ad platform may enable the auction participants to track a given user. Since the winning bidder (i.e., the ad selector) is the one serving the ad, our system can easily flag the corresponding page as being tracked by this bidder. The problem, however, is that we cannot ascertain if this ad selector could have received other bid requests for this user (while visiting other pages), and thus could have tracked them across those pages. Ad platforms typically permit bidders to build profiles *only* from the auctions they win, but, actually, nothing precludes them technically from exploiting such tracking data. In short, because there is no way of knowing the recipients of those requests and the use they make of such data, our knowledge of the sites tracked through RTB is limited to those sites where the ad selector serves an ad.

In this work, we address all such limitations by considering two scenarios in terms of tracking and sharing of clickstream data:

- a *baseline* scenario, where the system operates with the clickstream data that, according to our observations, the ad selector may have. That is, we assume that the observed clickstream of an ad selector matches the tracking data of which we are aware, and therefore we ignore any possible sharing of tracking information with other entities. In practical terms, our Web-browser extension will compile this clickstream by examining if the ad selector is present, as a third-party domain, on the pages visited by the user. In other words, we shall assume that all third-party domains present on a page may track a user's visit to such page. By doing so, we will be able to capture the sites where an ad selector has embedded a link (through the corresponding publishers), and those pages where it has won the right to serve an ad through RTB.

- a *paranoid* scenario in which we assume Web tracking is ubiquitous and clickstream information is shared among all entities participating in the ad-delivery process. In this case, we consider that the observed clickstream coincides with the actual clickstream, i.e., with the sequence of all pages a user has visited. We acknowledge, nevertheless, that there may not be ad companies and trackers on certain pages and thus a complete, accurate actual profile might not be captured in practice.

We would like to underline that the two scenarios described above refer *solely* to the user-tracking data available to ad selectors. Put differently, our system does not consider any interests data and personal information that users could have *explicitly* conveyed to these entities, and that could be utilized for ad-targeting purposes.

Having specified the two modes of operation of our system, next we introduce our second assumption, which concerns the way in which ad selectors construct user profiles from the observed clickstreams. In Sec. 5.3.2 we assumed that ad selectors model profiles as PMFs, essentially in line with a great deal of the literature on the field. To compute such distributions in practice, our system assumes, with a slight loss of generality, that ad selectors employ maximum-likelihood (ML) estimation [186]. We would like to stress that this is, by far, the most popular method of parameter estimation in statistics.

Our third and last assumption has to do with the topic categorization of the Web content. We shall consider that the categorizer used by our system coincides, to a large degree, with the one employed by ad platforms[13]. This implies that both our extension and ad platforms rely largely on the same predefined set of interest categories and the same categorization algorithm, so that any page visited by the user is classified into the same category by both the proposed system and the ad platforms tracking this visit. We believe this is a plausible assumption since our categorization algorithm builds on the standard topic taxonomy developed by the Interactive Advertising Bureau [13], an organization that accounts for the vast majority of online advertising companies in the US.

Components

This section provides a functional description of the main components of our prototype system architecture, justifies the design criteria, and gives *some* key, low-level implementation details. Fig. 5.5 depicts the implemented architecture, which consists of two main parts, the user side and the server side. The latter is in charge of computing the values of minimum uniqueness per ad selector. Because this requires obtaining $\bar{p}$, said computation is carried out only if the user accepts sharing their profile data with our server. The rest of functionalities and processing is conducted entirely on the user side. We analyze the components of both sides in the following subsections.

Profiles Estimator On the user side, this module aims at estimating (1) the set $\mathcal{P}$ of possible user profiles an ad selector may have assigned to a user; and (2) the distribution q of the interest categories of those ads classified as non-interest-based. It is important to stress that, regardless of the scenario assumed (i.e., baseline or paranoid), the estimation of q must be carried for *each* ad selector. In the former scenario, the computation of p is also necessary per ad selector. However, since the latter scenario considers that the observed clickstreams of all ad selectors match the user's actual clickstream, we just assume that $p = t$.

As explained in Sec. 5.3.2, the estimation of the PMF q requires a browsing session where the user is not tracked. Our current version of the plug-in implements this free-tracking session by means of the browser's private or incognito mode, a browser's feature that, among other functionalities, prevents tracking through HTTP and Flash cookies. We acknowledge, however, that ad selectors might also follow users' visits as a result of using super cookies, respawning [190, 36], canvas fingerprinting [160] or simply their IP addresses. Nevertheless, since these tracking mechanisms are either very infrequent or rather inaccurate, we may reasonably assume that the browser's incognito mode closely matches an untracked session, if not completely. In fact, recent studies indicate that the prevalence of these more sophisticated tracking methods is just 5% on top Alexa 100 000 sites [58]. In short, we shall therefore consider that the PMF q estimated this way effectively reflects the ad-topic distribution when the user is seen by the ad selector as a new user, and thus the ads can only be location-based, contextual and generic.

In practical terms, there is a difference between the estimation of p and q. In the latter case, it is conducted from the ads the browser receives during such incognito mode. In the case of p, or equivalently $\mathcal{P}$, the estimation is carried out from the pages the ad selector is able to track, on its own and/or through other sources of data.

One of the difficulties in estimating these two distributions is that, while q requires browsing in such free-tracking session, the PMF p must reflect the pages tracked by any potential ad selector. An approach to dealing with this incompatibility consists in alternating between the incognito and the normal modes on a regular basis. The problem with

[13] Ad platforms are the ones classifying the content of a page. In RTB advertising, they typically include the category of the publisher's page in the bid requests.

Table 5.1: Top-level interest categories.

adult	economics	hobbies & interests	politics
agriculture	education	home	real estate
animals	family & parenting	law	religion
architecture	fashion	military	science
arts & entertainment	folklore	news	society
automotive	food & drink	personal finance	sports
business	health & fitness	pets	technology & computing
careers	history	philosophy	travel

Table 5.2: Subcategories corresponding to three top-level categories.

Top-level category	**Bottom-level category**
arts & entertainment	animation, celebrities, comics, design, fine art, humor, literature, movies, music, opera, poetry, radio, television, theatre and video games.
health & fitness	alternative medicine, anatomy, asthma, autism, bowel incontinence, brain tumor, cancer, cardiac arrest, chronic pain, cold & flu, deafness, dental care, dermatology, diabetes, dieting, epilepsy, exercise, eye care, first aid, heart disease, HIV/AIDS, medicine, men's health, mental depression, nutrition, orthopedics, pediatrics, physical therapy, psychology & psychiatry, senior health, sexuality, sleeping disorders, smoking cessation, stress, substance abuse, thyroid disease, vitamins, weight loss and women's health.
personal finance	banking, credit, debt & loans, cryptocurrencies, financial news, financial planning, insurance, investing, retirement planning, stocks and tax planning.

such approach, however, is that users might be reluctant to browse in the private mode for the time needed to compute and update the PMFs q of a sufficient number of ad selectors.

Motivated by this, the user-side architecture simultaneously estimates both distributions by revisiting, in the incognito mode and in an automated manner, a *fraction* ρ of the pages browsed by the user. In practical terms, each revisit is made by opening a new minimized window in the private mode. We proceed this way because we want to avoid the tracking among different tabs in the same incognito mode. We admit, nonetheless, that this approach might have a non-negligible impact on these two aspects: first, in terms of the traffic overhead incurred; and secondly, it may penalize advertisers to some degree, since the ads received in the free-tracking session will obviously not be presented to the user. Currently, the proposed system operates with a revisit ratio of $\rho = 25\%$. Although this reduction in the number of revisits undoubtedly comes at the cost of inaccuracy in the estimation of q, we believe that it may account for an acceptable overhead in terms of traffic overhead and advertising impact. As a side note, we would like to stress that the impact of such revisits is, from a usability perspective, almost imperceptible.

After examining the Web-browsing conditions in which p and q are obtained, next we describe more concrete aspects related to the estimator of these distributions.

As mentioned in Sec. 5.4.2, this work assumes that ad selectors rely on ML estimation, a simple estimation method widely common in many fields of engineering. Let m denote the total amount of ads received (pages visited), and m_i the number of those ads (pages) which belong to the interest category i. Recall that the ML estimate of a PMF is defined as

$$q_i = \frac{m_i}{m},$$

for $i = 1, \ldots, n$.

In order to make a decision on whether the displayed ads are interest-based or not, our ML estimator requires observing the same minimum number of pages $w_{\min}$ needed by an

ad selector to model short-term interests. Several studies point out that the smallest time window that advertising companies might use for such modeling is one day (see Sec. 5.3.2). According to these studies and to the average number of pages browsed by a user per day [37], we set $w_{\min} = 87$. On the other extreme, in line with the works cited in that section, we consider that the largest clickstream used to model long-term interests is 8 weeks. We then set $w_{\max} = 3\,915$. To estimate q, we proceed analogously, by establishing a sliding window of this same length.

Lastly, on the server side, our architecture aims at computing, for each user willing to share profile data with the server, the average profile and the uncertainty class of each ad selector.

Web-Page Analyzer This block aims at obtaining certain information about (1) the Web pages browsed by the user and (2) the ads displayed within those pages, *both* in the tracked and in the incognito sessions. Specifically, when the browser downloads a page, being it in the normal or in the private mode, the module generates a list of all the entities tracking this page and serving ads on them.

In addition, our system attempts to retrieve the *landing page* of all ads displayed in both modes, that is, the page of the advertiser that the browser is re-directed to when clicking on its ad [131]. Recall that our system needs the interest category of an ad to make a decision on whether it is profile-based or not. In order to classify an ad into a topic category, the categorization module (described later in Sec. 5.4.2) requires its landing page. However, because clicking on every ad to get this information would lead us to commit click fraud [126], the functionalities provided by our tool in terms of transparency and blocking are limited to those ads where the landing-page information is available without clicking on them. Despite this limitation, some recent studies [131, 147] have reported an availability of the landing page above 80%.

Categorizer This module classifies the pages visited by the user as well as the landing pages of the ads directed to them, into a predefined set of topic interests. The module employs a 2-level hierarchical taxonomy, composed of 32 top-level categories and 330 bottom-level categories or subcategories. Tables 5.1 and 5.2 show the top-level categories and the subcategories corresponding to three of these categories.

The categorization algorithm integrated into our system is partly inspired by the methodology presented in [131] for classifying non-textual ads into interest categories. The algorithm also builds on the taxonomy available at the Firefox Interest Dashboard plug-in [45] developed by Mozilla.

Our categorizer relies on two sources of previously-classified data. First, a list of URLs, or more specifically, domains and hostnames, which is consulted to determine the page's category. Secondly, a list of unigrams and bigrams [152] that is used when the URL lookup fails. The former type of data is justified by the fact that a relatively small part of the whole Web accounts for the majority of the visits. Also, it is evident that pre-categorized lookup requires few computational resources on the user's browser and can be more precise. The latter kind of information, on the other hand, is justified as a fall-back and allows us to apply common natural-language heuristics to the words available in the URL, title, keywords and content.

For almost each of the top-level categories, the current version of the plug-in incorporates Alexa.com's 500 top Web sites. Also, the list of URLs includes the pages classified by Mozilla's plug-in (around seven thousand). On the other hand, the number of English unigrams and bigrams is approximately 76 000. Three additional lists, although of a fewer number of entries, are also available for French, Spanish and Italian[14]. To compile all these

[14]Upcoming versions of this Web-browser extension will include more languages.

words lists, we have built on the following data:

- a refined version of the categorization data provided by the Firefox Interest Dashboard extension;
- a subset of the English terms available at WordNet 2.0 [158] for which the WordNet Domain Hierarchy [151, 75] provides a domain label;
- a subset of the terms available at the WordNet 3.0 Multilingual Central Repository [119], to allow the categorization of Web sites written in the aforementioned languages;
- and the synset-mapping data between the versions 2.0 and 3.0 of WordNet [97].

The categorizer module resorts to these lists only when the hostname and domain are not found in the URL database. When this happens, the algorithm endeavors to classify the page by using the unigrams and bigrams extracted from the following data fields: URL, title, keywords and content. Depending on the data field in question, the categorizer assigns different weights to the corresponding unigrams and bigrams. In doing so, we can reflect the fact that those terms appearing in the URL, the title, and especially the keywords specified by the publisher (if available), are usually more descriptive and explanatory than those included in the body of the page.

As frequently done in information retrieval and text mining, our Web-page classifier also relies on the term frequency-inverse document frequency (TF-IDF) model [183]. Said otherwise, we weight the resulting category/ies based on the frequency of occurrence of the corresponding unigrams and bigrams, and on a measure of their frequency within the whole Web.

Optimization library	**Running time** [s]		
	average	variance	maximum
Coin-OR Linear Programming (CLP), v.1.16.6 [6, 148]	0.0315505	0.0000010	0.0460014
GNU Linear Programming Kit (GNULPK), v.4.48 [12]	0.0337618	0.0000055	0.0681626
Object Oriented Quadratic Programming (OOQP), v.0.99.22 [115]	0.0401395	0.0000028	0.0805860
LPSolve, v.5.5.2.0 [76]	0.0645488	0.0000024	0.0808482
C Library for Semidefinite Programming (CDSP), v.6.1 [81]	0.5878725	0.0017888	1.1033131
Dual-Scaling Semidefinite Programming (DSDP), v.5.8 [74]	2.0933280	0.0137100	4.1946620
Coin-OR Interior Point OPTimizer (IPOPT), v.3.12.3 [5, 201]	0.2014676	0.1510803	5.7872007
Limited Memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS), v.3.0 [209]	0.2054921	0.1669853	6.1828331
NLopt, v.2.4.2 [130]	0.5781220	0.0010485	0.6520662

Table 5.3: We tested 6 optimization libraries to compute the solution to the LP problem (5.3), and another 3 for the divergence-minimization problem (5.4). This figure shows the average, the variance and the maximum values of running time, obtained from one thousand problem instances. Each solver is listed along with the corresponding version number.

For the sake of computational efficiency, the algorithm stores the categories derived from the user's last 500 visited pages. This way, when the user re-visits one of those pages, the topic categories are obtained directly without needing to go through the process above.

In terms of storage, the whole list of unigrams, bigrams and their corresponding IDF values occupies approximately 1 megabyte in compressed format. We believe this is an acceptable overhead to the plug-in download size.

Lastly, a manual inspection of the categorization results for a large collection of Web pages and ads indicates that the algorithm is, in almost all cases, certainly precise. Further investigation would be required, however, to evaluate the performance of the categorizer in a more rigorous manner.

Optimization Modules The optimization modules incorporated in the user side and the server side are responsible for computing the solutions to the problems (5.3) and (5.4), and thus obtaining the robust minimax detector and the minimum profile uniqueness, respectively. The input parameters of the user-side module are the distribution q and the tuples $p^{\min}$ and $p^{\max}$. On the server side, our system requires the observed clickstream of each ad selector to compute the average profile and the associated uncertainty class. We would like to remark that the ad transparency and blocking functionalities related to profile uniqueness will only be provided should the user consent to convey such clickstream data.

In the architecture implemented, both modules rely on open-source optimization libraries. The design of such modules required the examination and comparison of a variety of optimization solvers to this end. Because our system may need to compute the robust detector each time an ad is displayed, we endeavored to prioritize efficiency and reliability on the user side. These same requirements were also allowed for on the server side. However, because the minimum-uniqueness values $u_{\min}$ are meant to be computed for *each user*, we opted to lighten the processing and computation in this part of the architecture. Particularly, instead of processing the profile data every time there is an update on the user side, we specify regular intervals of 1 day (from the time the plug-in is installed) for the exchange of information with the server. We acknowledge that, depending on the user activity, this might have a certain impact on the accuracy of the profile-uniqueness data provided.

With all these requirements in mind, we performed a benchmark analysis for the LP and divergence-minimization problems. We employed the Matlab optimization toolbox OPTI [94], and tested one thousand problem instances with random —although feasible— values for the inputs mentioned above[15]. For the problem (5.4), and when available at the optimization software under test, we also provided the gradient and the Hessian of the objective and constraint functions. In addition, we reduced the complexity of this latter problem by using a top-level representation of $\bar{p}$, $p^{\min}$ and $p^{\max}$ with only 32 categories.

The results are shown in Table 5.3 for 9 optimization solvers. Based on our performance analysis, we selected the CLP [6, 148] and IPOPT [5, 201] libraries, which provide a simplex and an interior-point method [82], respectively. The two solvers exhibited the lowest average running time in our analysis, with 32 and 201 milliseconds respectively, as well as acceptable variance values. It is worth mentioning that all problem instances were solved satisfactorily by the libraries tested, and that the two solvers chosen are available under the Eclipse Public License [10].

In our system, both solvers were configured to have a maximum allowable running time. When our extension is installed for the first time, it runs several problem instances to set this parameter; this is for the computation of the robust minimax interest-based ad detector. On the server side, the computation of the minimum value of user-profile uniqueness is limited to 0.5 seconds.

Blocking Policies The functionality of this module is to apply the ad-blocking policies defined by the user. Its current implementation simplifies the formal policy notation presented in Sec. 5.4.1, in an attempt to provide an easy-to-use interface and thus enhance usability.

With this aim, our extension allows users to define policies *only* with negative *sign*. That is, instead of specifying which ads should be displayed (+) and which ones should be blocked (-), we just enable the latter blocking declaration, which may facilitate the definition of such policies. In addition, the specification of percentile values of profile-

[15]The optimization software was tested on an Intel Xeon E5620 processor, equipped with 8 GB RAM, on a 32-bit Windows 7 operating system.

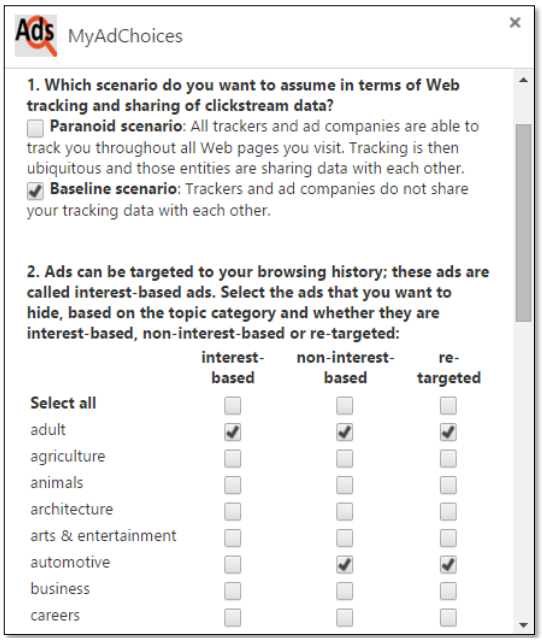


Figure 5.6: The configuration panel shown in this figure allows users to define fine-grained, ad-blocking policies. The options available to users include filtering out ads per interest category, behavioral and retargeting advertising. Although not displayed in this figure, users can also denote ad-blocking conditions depending on the uniqueness of the profiles that ad selectors might potentially build.

uniqueness is, in this implementation, reduced to a binary choice: users can only decide if they wish to block (or allow) those entities which may have compiled "very unique" profiles of them, meaning that $\pi_{u_{\min}} \geqslant 90\%$. Fig. 5.6 shows the configuration panel by which users may configure blocking policies, as well as the scenario they wish to assume in terms of Web tracking.

The operation of this module is described next. When a user visits a page, the module waits for the categorizer to send the topic category of each ad to be displayed. Then, it receives the robust minimax interest-based ad detectors of each of the entities delivering those ads. And finally, it consults an internal database (i.e., on the user side) to obtain the minimum uniqueness values associated with such entities. With all this information, our system only needs to verify if each ad constraint is satisfied and, accordingly, decide whether to block the ad or not.

We must highlight that our system does not block the ads in the same sense as current ad-blocking technologies do. While these technologies prevent third-party network requests[16] from being sent, our Web-browser extension does allow them. It is only when the page is completely loaded and thus the ads (if any) are displayed, that our system decides to hide them or not by applying a black mask on top of them[17]. To highlight this particular aspect, we refer to the action of blocking more precisely as *hiding* or *obfuscation*. Fig. 5.7 shows a screenshot of the ads processed by our tool in a particular Web page.

The tool notifies users about the kind of ads received through a small icon placed on the left corner of each detected ad. The icons indicate if an ad is interest-based (red), retargeted (red), non-interested-based (green), it is blocked according to the user's policy (black), or the system cannot make a decision (orange). This latter case occurs, for example, when

[16] AdBlock Plus [2], for example, do not block *all* third-party network requests but only those blacklisted [19].

[17] On a technical note, the system might alternatively remove the ad image.

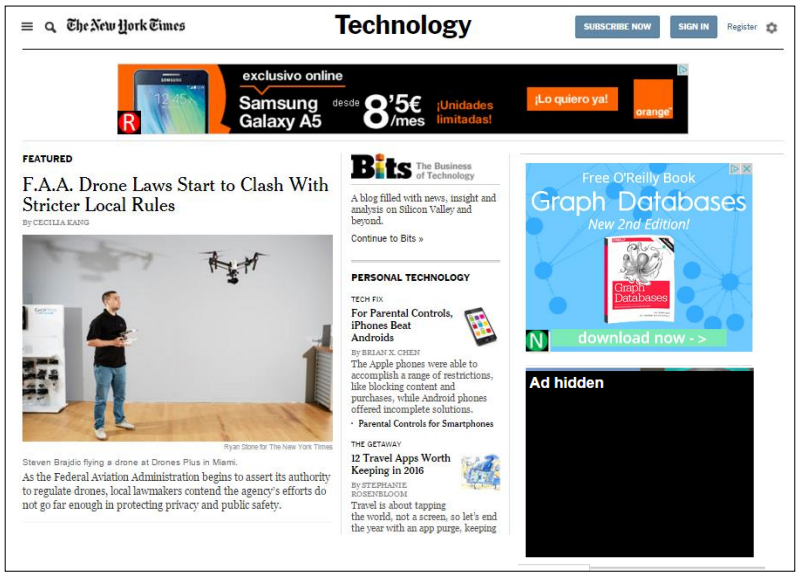


Figure 5.7: We show a screenshot of the ads identified by our system in The New York Times' Web site. One of these ads is classified as retargeted, another as non-interest-based, and the bottom-right one is hidden according to the user's blocking policy.

the ad's landing page is not available or the categorizer cannot classify it; when there is insufficient data to train the PMF models of p and q; or when the execution of the optimization solver exceeds the maximum allowable running time.

5.5 Evaluation

In this section, we empirically evaluate the proposed system and analyze several aspects of behavioral advertising. The analysis of this form of advertising is conducted from the ads as well as browsing data of 40 users of *MyAdChoices*. To the best of our knowledge, this study constitutes the first, albeit preliminary, attempt to investigate behavioral targeting and profile uniqueness in a real environment from real user browsing profiles.

5.5.1 Data Set

We distributed *MyAdChoices* to colleagues and friends and asked them to install it and browse the Web normally for one month. The experiment was conducted from December 2015 to January 2016. The data collected by our Web-browser extension were sent to our servers every one hour. On the other hand, the extension was configured for a fraction of revisited pages of 100%. That is, every page browsed by a user was revisited by our system in the incognito mode.

The participants were mostly researchers and students based in our countries of residence, France, India and Spain. No attempt was made to link the gathered data to the personal identities of the volunteers. As a preprocessing step, we removed those users who visited less than 100 sites, leaving a total of 40 users.

5.5.2 Results

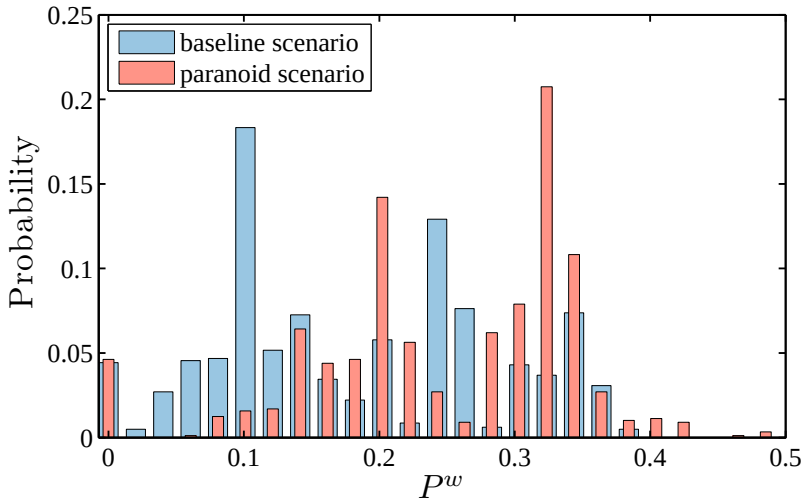


Figure 5.8: PMF of the worst-case error probability for the two scenarios assumed in this work.

System Performance

Evaluating an ad-transparency tool is extremely challenging since the ground truth of targeting decisions is unknown. The effectiveness of these tools has been occasionally assessed through manual inspection [140, 96]. However, this approach has been recently shown to be extremely prone to errors [141]. In this section, we evaluate the error probability of the interest-based ad detector bearing in mind the impossibility of checking a detector's decisions with the true condition of the tested ads (i.e., whether they are actually interest-based or not).

Before proceeding with this evaluation, we first report the availability of categorization data in our data set. Recall that our system classifies ads into topic categories from their landing pages. To this end, the categorization module makes use of the words included in the landing page's URL, keywords, title and content. In our series of experiments, we found that just 0.60% of ads could not be categorized by using this information, which represents a good availability index. In most of the cases, the reason was the lack of language support. As explained in Sec. 5.4.2, currently our categorization module works only for English, French, Spanish and Italian.

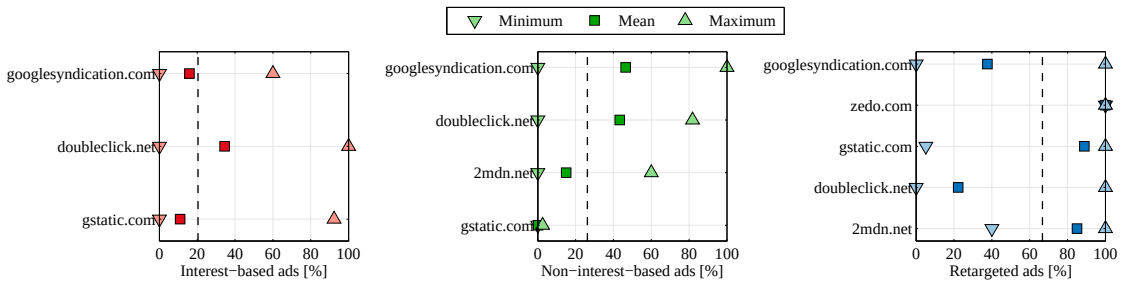


Figure 5.9: *Ad selectors.* Interest-based, non-interest-based and retargeted ads for the *baseline* scenario.

Having checked the performance of our categorizer, now we turn to the robust minimax detector. In all the executions of the optimization library CLP (including both the baseline and the paranoid scenarios) no single error was reported to our servers. That is, our system was able to successfully compute said detector, without exceeding the maximum allowable running time for this computation, set to 0.5 seconds in these experiments. Likewise,

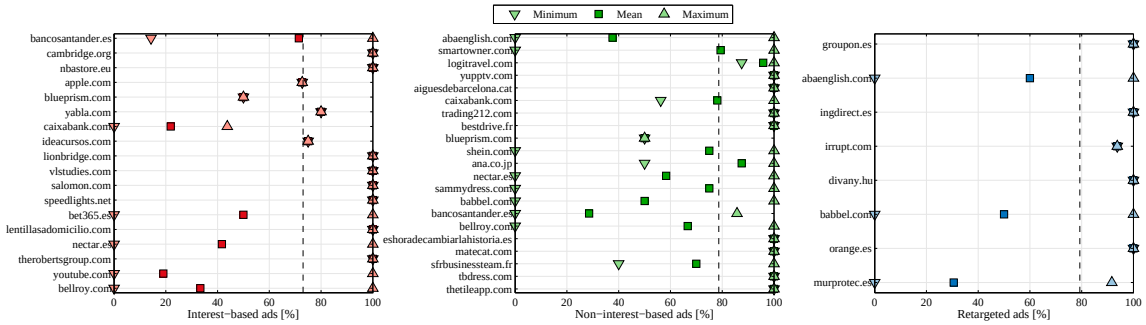


Figure 5.10: *Advertisers.* Interest-based, non-interest-based and retargeted ads for the *baseline* scenario.

the IPOPT software did not report any error when computing the values of minimum uniqueness.

Fig. 5.8 shows the PMF of the probability of error of the interest-based ad detector. In the baseline scenario, we observe a mean and a variance of 0.1827 and 0.0105, respectively. In the paranoid case, these two moments yield 0.2504 and 0.0094. Two remarks are in order from these figures. First, both cases exhibit relatively low error probabilities, with expected values roughly lower than $1/4$. Secondly, the paranoid scenario seems to be slightly more prone to errors in terms of interest-based ad detection. One possible explanation for this is a greater semblance between the distributions p and q in this scenario. Intuitively, the more dissimilar these distributions are, the lower is the probability of incorrectly identifying an interest-based ad.

Table 5.4: Minimum, mean and maximum percentage values of interest-based, non-interest-based and retargeted ads over all users in our data set.

	Baseline scenario [%]			**Paranoid scenario** [%]		
	min.	mean	max.	min.	mean	max.
Interest-based	0	13.2	60.0	0	17.8	66.7
Non-interest-based	0	31.7	78.4	0	29.4	76.1
Retargeted	0	55.1	100	0	52.8	100

Behavioral and Retargeted Advertising

This section examines several aspects of behavioral advertising and retargeting, including an analysis of the entities delivering such forms of advertising; the topic categories most targeted in our experiments; the discrepancy between the baseline and paranoid scenarios; and a preliminary study of the relationship between interest-based advertising and profile uniqueness.

Some general figures on behavioral and retargeted advertising are shown in Table. 5.4. To obtain these figures, we computed, for each user with a minimum of 10 ads received, the percentage of interest-based, non-interest-based and retargeted ads. The minimum, mean and maximum values of those percentages over all users are the values represented in this table.

The results clearly indicate that retargeting is the most common ad-targeting strategy, followed by non-interest-based advertising and behavioral targeting. This order is observed both in the baseline and in the paranoid scenario, with small differences in the percentage values. One of the most interesting results is the relatively small prevalence of behavioral targeting, which accounts for one third of retargeted ads. This is in contrast with previous

work reporting higher average percentages of this type of advertising for fake profiles [86], but in line with recent marketing studies [67] which point out that retargeted ads are preferred to interest-based ads in a proportion 3:1.

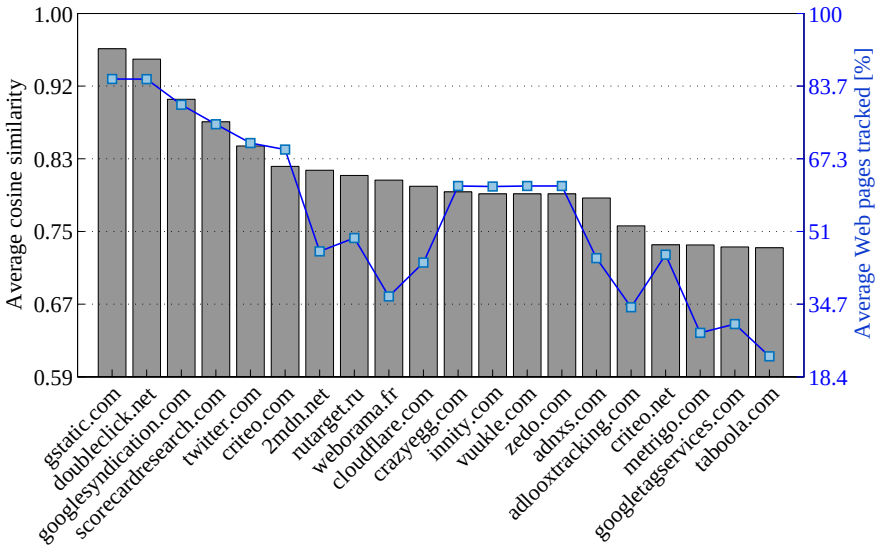


Figure 5.11: We show the cosine-similarity values between the actual and the observed profiles, averaged over all users and per tracking entity.

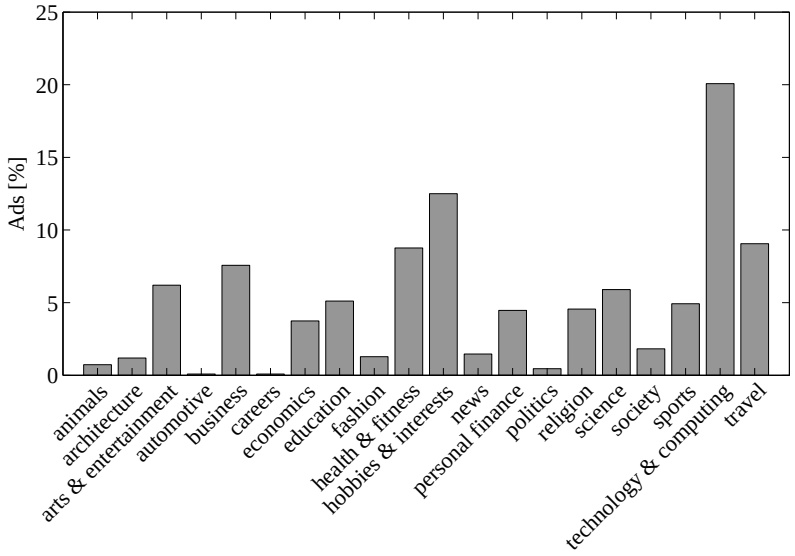


Figure 5.12: Percentage of ads across the top 20 topic categories.

Ad Selectors and Advertisers In this subsection, we examine the ad selectors which, in our data set, were responsible for the delivery of behavioral, non-behavioral and retargeted advertising. We computed, to this end, the percentage of interest-based, non-interest-based and retargeted ads served by each of these entities. Fig. 5.9 depicts the minimum, mean and maximum values of such percentages for each ad selector delivering a minimum of 10 ads; these results correspond to the baseline scenario. In each of the three diagrams, ad selectors were sorted in decreasing order of total number of served ads, from top to bottom. The dot vertical lines indicate average percentages over the ad selectors displayed.

The figure in question shows only five ad selectors. In our data set, these entities were responsible for 98.99% of the total number of ads. Not entirely unexpectedly, Google's ad companies (`googlesyndication.com`, `doubleclick.net` and `gstatic.com`) were the ones monopolizing the three ad classes. The former ad platform was observed to target mostly non-interest-based and retargeted ads, whereas DoubleClick and `gstatic.com` focused on behavioral advertising and retargeting, respectively. The remaining ad selectors were `zedo.com` and `2mdn.net`. The majority of ads served by these ad companies were retargeted. Lastly, the paranoid case exhibits similar results and is omitted for the sake of brevity.

The same methodology was used to analyze the advertisers of our data set, and to generate Fig. 5.10. This figure shows Banco Santander, Cambridge University Press, NBA Store and Apple as the advertisers with the highest rates of behavioral advertising. SmartOwner, Logitravel.com, YuppTV and CaixaBank, on the other hand, lead the ranking of non-interest-based ads, and Groupon, ABA English and Ing Direct are the companies most interested in retargeting. Although we cannot derive a general rule from these results, we note that large companies are more frequent in the behavioral-targeting list than in that of non-interest-based ads. This might be an immediate consequence of the higher chances of such firms —for example— to win ad-auctions at RTB, compared to companies with limited purchasing power.

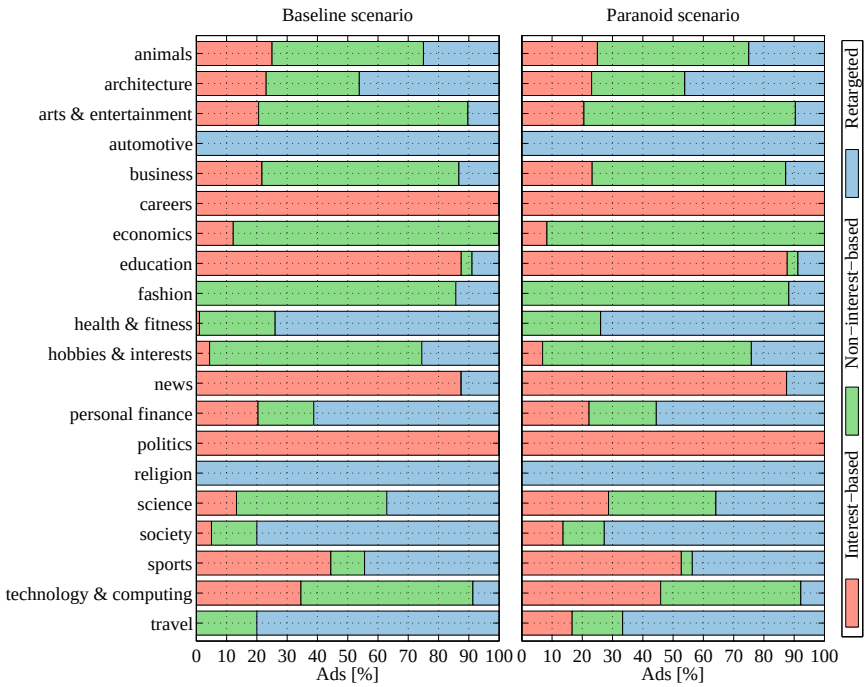


Figure 5.13: Some of the top-level interest-categories targeted in the baseline and the paranoid scenarios.

Baseline and Paranoid Scenarios Next, we analyze the overall percentage of coincidence between the baseline and the paranoid scenarios in terms of interest-based ad detection. To this end, for each ad we checked if the decision made by the detector in the baseline mode matched the decision made by the detector in the paranoid case.

The percentage of matching observed in our data set was certainly high, especially for the ad platform `gstatic.com`, which yielded 97.4%. Although smaller, the percentages of coincidence for DoubleClick (75.6%) and `googlesyndication.com` (87.0%) were also remarkable. A plausible explanation to this behavior is the semblance of the profile p

estimated in both scenarios, which might indicate that `gstatic.com` relied only on its own tracking data and thus did not enrich this information with browsing profiles from other sources.

Precisely, the semblance of the profiles p and t is investigated in our next figure, Fig. 5.11. Recall that these profiles are estimated from the observed and the actual clickstreams respectively. To compute Fig. 5.11, we kept a record of all entities tracking users' visits; these entities were ad platforms, advertisers and also data-analytic trackers. Then, from said records, we calculated the percentage of pages tracked by each of these entities, as well as the cosine similarity between the observed and actual profiles. The figure at hand shows these percentage and similarity values averaged over all users.

A couple of remarks follow from this figure. First, Google's ad platforms are the entities with the most extensive tracking capabilities. Particularly, `gstatic.com`, DoubleClick and `googlesyndication.com` tracked users on 92.9%, 88.2% and 81.2% of the visited pages. An immediate consequence of this are the high values of cosine similarity observed. Secondly, the results are consistent with the percentages of scenario matching provided at the beginning of this subsection. Thirdly, the profiles p of ad companies with limited tracking capabilities like Metrigo and Taboola were observed to be relatively similar to the corresponding actual profiles. Although it is not possible to find an accurate answer for this result, the reason might be found in the model of user profile based on *relative* frequencies.

Finally, we would like to emphasize the appropriateness of the proposed scenarios for the particular ad selectors examined in these experiments. Recall that the baseline scenario does not contemplate the sharing of tracking information with other ad selectors and trackers, whereas the paranoid case does; this latter scenario also considers that tracking is ubiquitous. The results provided throughout this experimental section build on the assumption that `googlesyndication.com`, DoubleClick and `gstatic.com` operate independently in the baseline scenario. However, since they are all Google ad companies, one might expect that these three firms would have exchanged information with each other. The paranoid scenario precisely captures this possible exchange of tracking data. Also, the ubiquitousness of tracking is justified by the fact that these ad platforms combine for a total of 99.08% pages tracked (i.e., they track users almost in all pages they visit).

Interest-Categories Targeted Fig. 5.12 plots the probability distribution of the ad-topic categories. In this figure, we considered only those topics for which we collected a minimum of 5 ads. The results indicate that the most popular interest categories were "technology & computing", "hobbies & interests", "travel" and "health & fitness", with percentages of 18.4%, 11.5%, 8.3% and 8.1%, respectively.

Fig. 5.13 illustrates, on the other hand, the targeting strategies that were observed in each of the 20 categories represented in Fig. 5.12. As can be seen, very similar results were reported for the baseline and the paranoid scenarios. Our findings show that retargeted ads were more frequent on categories like "automotive", "religion", "society" and "travel", which seems to be partly in accordance with some marketing surveys [111, 83]. On the other hand, profile-based ads were observed more predominantly on "careers", "education", "news" and "politics", and non-interest-based ads were largely targeted to "fashion", "economics" and "hobbies & interests".

Behavioral Targeting and Profile Uniqueness In our last experiments, we briefly explore whether common browsing profiles are more likely (or not) to receive interest-based ads. With this purpose, for each ad classified as interest-based and non-interest-based, we analyzed the minimum-uniqueness values of the ad selector serving it. The probability distributions of such values are plotted in Fig. 5.14.

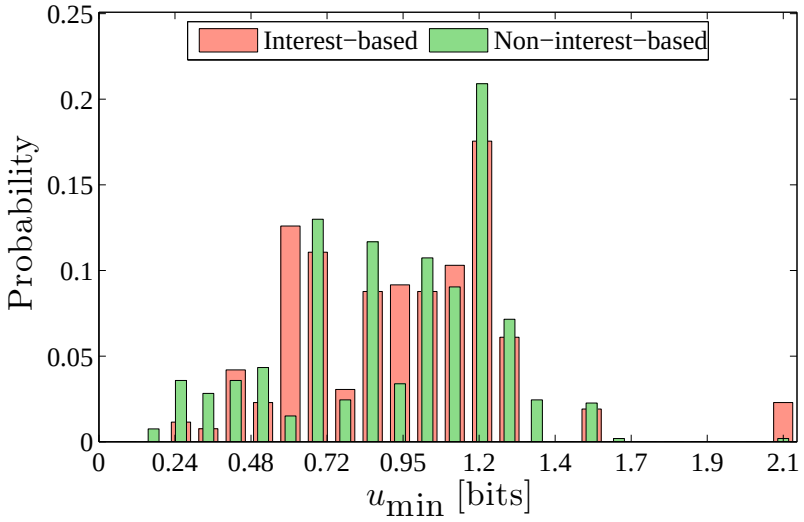


Figure 5.14: We show the PMFs of the profile-uniqueness values analyzed, when ads are classified as interest-based and when they are considered to be non-interested-based.

As can be observed, the two PMFs are very similar, which clearly means that the probability of delivering an interest-based ad may not depend on the uniqueness of the observed profile. In fact, the expected values of these distributions are 0.8949 bits for profile-based ads, and 0.8834 bits for non-interest-based ads; and the KL divergence (a measure of their discrepancy) yields 0.4344 bits. On the basis of the evidence currently available, it seems fair to suggest that the uniqueness or commonality of a profile is not a feature that ad selectors in general use to decide their user-targeting strategies. Further evidence supporting this assertion, however, would require the analysis of larger volumes of data.

5.6 Related Work

This section reviews the state of the art relevant to this work. We proceed by exploring, first, the current software technologies aimed at blocking ads; and secondly, we examine those approaches intended to provide transparency to online advertising.

5.6.1 Ad Blockers

The Internet abounds with examples of ad-blocking technologies. In essence, these technologies act as firewalls between the user's Web browser on the one hand, and the ad platforms and tracking companies on the other. Specifically, ad blockers operate by preventing those HTTP requests which are made when the browser loads a Web page, and which are not originated by its publisher. These requests are commonly referred to as third-party network requests, as mentioned in the introductory section of this work.

Most of these tools are implemented as open-source browser plug-ins, and carry out said blocking with the help of a data base or *blacklist* of ad platforms and trackers. Basically, these lists include regular expressions and rules to filter out the third-party network requests that are considered to belong to ads or trackers. The maintenance of such blacklists is done manually by the technologies' developers and in some cases by user communities. Some of the most popular ad-blockers are Adblock Plus [2] and Adblock [122]. Within this list of blocking technologies, we also include anti-tracking tools like Ghostery [11], Disconnect [8], Lightbeam [15] and Privacy Badger [22], which, from an operational point of view, work exactly as ad blockers and thus may as well block ads.

A middle-ground approach for ad-blocking has recently emerged that uses whitelists to allow only "acceptable ads". The criteria for acceptability typically comprise non-invasiveness, silence and small size [185]. However, because these criteria ultimately depend on the ad blockers' developers, this approach does not signify any real advance in the direction of returning users control over advertising. Indeed, this "acceptable-ads" approach has caused a great controversy in the industry, when it came to the public that the most popular ad blocker was accepting money from some of the whitelisted companies [90].

Table 5.5: Comparison between MyAdChoices and other tools that may provide transparency to behavioral advertising.

Approaches	**Type of tool**	**Disadvantages**
[86, 85]	research platform	◦ valid for single-category profiles, ◦ transparency functionality available only on weather pages, ◦ inaccurate model of the ad-delivery process, ◦ parallel browsing in incognito mode, ◦ only paranoid scenario, ◦ multiple user-targeting objectives not allowed;
[73]	research platform	◦ valid for single-category profiles, ◦ transparency functionality limited to users visiting the same pages, ◦ generic and contextual ads are omitted, ◦ only paranoid scenario;
[147]	research platform	◦ only for DoubleClick, ◦ simplified model of the ad-delivery process (e.g., generic ads and RTB ignored, only for long-term user profiles), ◦ binary decision, i.e., ads are either contextual or interest-based, ◦ inconsistent model of tracking and sharing of user data;
[140, 141, 96]	research platform	◦ not scalable for Web browsing [140, 141], ◦ unacceptable network traffic and computational overhead, if intended for end users [140, 141], ◦ may detect retargeting but not behavioral advertising;
MyAdChoices	end-users tool	◦ a fraction of revisits in incognito mode.

5.6.2 Advertising Transparency

To the best of our knowledge, in terms of transparency, our work is the first to provide *end-users* with detailed information about behavioral advertising in *real-time*. As we shall see next, only a couple of previous works tackle the problem of interest-based ad detection. The major disadvantage of these few existing approaches, however, is that they are not intended for end-users, i.e., they are not designed to be used by a single user who wishes to find out what particular ads are targeted to them. Instead, these approaches consists in platforms aimed at collecting and analyzing advertising data at large scale for *research* purposes. In general, they allow running experiments in a limited and controlled environment, and studying the ads displayed to very specific and artificially-generated user profiles.

In this subsection we shall examine these works, bearing in mind that none of them are conceived as a tool that users can directly and fully benefit from it. In addition, and equally importantly, we shall see that these proposals rely on a too simplistic, and in many cases erroneous, model of the actual ad-delivery process. Also, they very often resort to simple heuristics, not rigorously justified, to conduct their measurement studies on behavioral advertising.

In contrast to these works, we propose a formal study of this form of advertising that builds on a more general, accurate model of the ad-serving process, which takes into account its complexity and the new paradigm of RTB, and which addresses the challenges

others simply neglected. We proceed by following a mathematically grounded methodology that capitalizes on the fields of statistical estimation and robust optimization. Besides, compared to these works, our analysis of behavioral targeting does not only determine if an ad is interest-based or not, but also it explores a crucial aspect of the interests tracked and profiled by ad companies, namely, the commonality of user profiles. Next, we elaborate more on these proposals.

The first attempt to identify the challenges that may arise when measuring different aspects of online advertising was done in [121]. Although not particularly interested in behavioral targeting, the authors investigated aspects like the impact of page reloading and cookies on advertising, and highlighted the difficulties found through some simple experiments.

Following this work, [86, 85] proposed a platform which automatizes the collection of certain statistics about behavioral targeting. The proposed platform creates artificial user profiles with very specific, non-overlapping topic categories (i.e., profiles with active categories only in sports, only travel, and so on) by emulating the visits to pages related to those topics. The tool in question alternates this training browsing with visits to weather Web pages, where they check if the categories of the received ads match the category of the corresponding profile; the authors justify the use of these weather-related pages by arguing that, there, contextual ads are detected more easily. To carry out this checking, the tool first filters out those landing pages which may correspond to generic and contextual ads. With this aim —and similarly to our tool—, it revisits, in incognito mode, each visit to a weather page and keeps a record of the ads delivered in this session. By eliminating the landing pages common to both sessions, the authors claim to discard the *majority* of untargeted and content-based ads.

Apart for the fact that said platform is not intended for end-users nor provides real-time ad-transparency functionalities, the most important drawback is its extremely limited scope of application. First, it only works for single-interest profiles, and secondly, transparency can only be brought in such weather pages, which provides very simplistic and superficial insight into behavioral targeting. Nonetheless, this is not the only limitation. To detect interest-based ads, the authors make the mistake of oversimplifying the ad-delivery process by assuming some sort of determinism: they consider that *most* of the non-interest-based ads a user may receive in a tracked and in a free-tracked session will be exactly the same, which totally neglects the inherent randomness of the ad-serving process. Besides, the authors do not consider the particular ad platform serving an ad and therefore implicitly assume —although they do not mention it— a worst-case or paranoid scenario in terms of tracking and sharing of data. This is in contrast to our work, which in addition considers a baseline scenario for tracking.

Finally, the cited works [86, 85] evaluate their approach by using a distance measure between the terms appearing in the ads' landing pages and those in the training pages. While this quantifies the similarity between the ads' topic categories and profiles' single categories, the authors do not assess the method to detect profile-based ads. An important consequence of this lack of evaluation is that generic ads belonging to the profile's active category will always be classified as interest-based (provided that they have not been delivered in the incognito sessions), and the platform will not report any error on this classification. On the contrary, MyAdChoices provides, for each ad, the probability of error incurred in estimating its class.

Following the same spirit, [73] presents an ad-crawling infrastructure that does not aim exactly to provide transparency, but to analyze different aspects of advertising at large scale. Among other aspects, the authors study the average arrival rate of new ads and the distribution of the number of ads and advertisers per page. In addition, they briefly examine behavioral targeting by following a similar approach to that of [86, 85]. They

emulate the browsing habits of around 300 users with single-category interests, and try to see which ads are more targeted to which profiles when visiting a subset of selected Web pages. Their analysis of profile-targeting assumes that, if an ad is shown more frequently to a given profile than to others, then this ad is targeted to said profile. Building on certain heuristics, the authors compare the frequency of appearance of each ad (for each profile) with a uniform profile, and conclude that an ad is targeted if the result of such comparison exceeds a certain threshold.

The proposed framework suffers from the same limitations of the aforementioned two works. Besides, the authors disregard that ads can be contextual and generic as well, and that the frequency of appearance of ads depends on highly dynamic factors. On the other hand, a practical implementation of this framework on the user side would be unfeasible as it would require that users visit the same pages (to enable the transparency functionality), and exhibit single-category profiles.

A similar platform is proposed in [147] that studies the ads delivered to some artificial profiles, in this case built from the AOL search query data set [69]. The tool is not intended for end-users and provides a framework that aims to study interest-based and contextual advertising at large scale. The platform, which operates offline and is restricted to DoubleClick ads, analyzes two data sets to this end: the interest categories of *all* ads received both in a tracked session and in an incognito-browsing mode. The authors then use a binary classifier to decide if an ad belonging to a certain category is interest-based *or* contextual.

The major limitations of this tool come from the simplified and inaccurate model assumed for the ad-delivery process. First, it does not take into account generic or untargeted ads. Secondly, the decision is binary in the sense that the result of an ad classification cannot be contextual *and* interest-based, thus overlooking that the vast majority of ad platforms allow the selection of multiple user-targeting objectives. Thirdly, such classification relies on the *whole* data set of ads collected in the tracked and incognito sessions, which neglects the fact that DoubleClick (as any ad selector) may construct short-term profiles or use *any* time window to profile users' browsing interests, not necessarily the one that spans the whole browsing history. Last but not least, the tool in question does not reflect the actual operation of the ad platform it focuses on, namely, that DoubleClick may employ modern RTB technologies to serve ads [165, 164]. On the one hand, the authors seem to assume a baseline scenario, as the user profile is built just from the pages tracked by this ad platform. But on the other hand, they completely ignore the RTB ad-serving technology, and the fact that DoubleClick's ad-auction participants may not share the same profiling data. That is, the authors seem to assume, at the same time, a paranoid scenario, which is contradictory. We would like to stress that our work addresses all these four issues, by modeling the combination of multiple ad-targeting decisions, relying on the notion of ad selector, building independent user-profile models per ad selector, and considering any possible time window chosen by such entities through the definition of uncertainty class.

Another more recent work for conducting experiments based on artificial profiles is [140], which tracks the personal data collected by several Web services, and tries to correlate data inputs (e.g., e-mails and search queries) with data outputs (e.g., ads and recommended links). The proposed platform tackles this correlation problem in a broad sense, and is tested for the ads displayed on Gmail. The platform relies on the maintenance of a number of *shadow accounts*, that is, replicates of the original account (e.g., an e-mail account), but which differ in a subset of inputs. All these account instances are operated in parallel by the system and are used to compare the outputs received. Intuitively, if an ad is displayed more frequently on those accounts sharing a certain input (e.g., an e-mail), and this ad never shows up in the rest of shadow instances, then this input is likely to be the cause of said ad.

The platform in question does not require a shadow account for each possible combination of input data, but a logarithmic number of such accounts in the number of inputs, which makes it suitable for the application where it is instantiated. However, it would be totally infeasible to extend it so as to analyze the ads received out of this controlled application, for example, while browsing the Web. First, in terms of scalability. The authors claim to support the correlation of hundreds of inputs (e-mails), with reasonable costs in terms of shadow accounts. This may work for a *single* service provider, but clearly not when considered in the more general context of Web advertising, with thousands of ad companies tracking users throughout the Web [11] and around ninety pages visited on average per day [37]. Secondly, creating equivalent shadow browsing profiles on the user side would be impractical in terms of network traffic and computational overhead. On the other hand, the proposed solution checks which particular input data or combination (with a reduced combination size, to attain the scalability mentioned above) is responsible for a given output data (e.g., an ad). As a result, such platform may work for advertising forms like retargeting, where a single visit may be the cause of an ad display, and for contextual ads, which depend on the page currently being visited. However, it does not operate on a much coarser granularity level and hence it is not suitable for studying behavioral targeting, where ads are typically served on the basis of browsing histories accumulated over long time periods.

A couple of refinements of this latter approach are [141, 96], which respectively provide certain statistical validation of its findings and which investigate causation in text-based ads. The cited works, however, are measurement platforms and suffer from the same limitations in terms of detecting behavioral targeting in a broad sense. Table 5.5 summarizes the major conclusions of this section.

5.7 Conclusions and Future Work

In the last few years, as a result of the proliferation of intrusive and invasive ads, the use of ad-blocking and anti-tracking tools have become widespread. The problem with these technologies is that they pose a binary choice to users and thus disregard the crucial role of advertising as the major sustainer of the Internet's free content.

We believe that such technologies are only a short-term solution, and that better tools are necessary to solve this problem in the long term. Most users are not against ads and are actually willing to accept some ads to help Web sites. However, this is provided that the ad-delivery process be transparent and users can control the personal information gathered.

Since different users may have different motivations for using ad blockers and anti-trackers, this chapter proposes a smart Web technology that can bring transparency to online advertising and help users enforce their own particular choices over ads. The primary aim of this technology is, first, to let users know how their browsing data are exploited by ad companies; and secondly, to enable them to react accordingly by giving them flexible control over advertising.

The proposed technology provides transparency to behavioral targeting by means of two randomized estimators. The former builds on a theoretical model of the ad-serving process, and capitalizes on the methodology of robust optimization to tackle the problem of modeling the profiles available at ad platforms. The latter sheds light on these profiles by computing a worst-case uniqueness estimate over all possible profiles constructed by an ad platform.

These two detectors have been integrated into a system architecture that is able to provide ad transparency and blocking services all in real-time, and on the user side. In terms of transparency, our tool enables users (1) to learn if the ads delivered to them may

have been targeted on the basis of their browsing profiles, and (2) to find out whether such profiles may be revealing unique browsing patterns. In terms of ad blocking, the proposed system allows users to filter out interest-based, non-interested-based and retargeted ads per topic category, and to specify blocking conditions based on profile uniqueness.

The proposed system has been implemented as a Web-browser extension and assessed in an experiment with 40 participants. In terms of performance, the two estimators exhibited running times below 0.5 seconds and reported no errors. In addition, nearly all pages could be categorized. We carried out an analysis of behavioral targeting based on the ads and browsing data of those volunteers. Among other results, our findings show that retargeting is the most common ad-targeting strategy; that GoogleŠs ad companies are the ones leading behavioral and retargeted advertising; that large firms might be the advertisers mostly delivering profile-based ads; and that profile uniqueness may not be a widely used criterion to serve ads.

Unlike few previous work on Web transparency, our tool is intended for end-users, departs from a more faithful, accurate model of the ad-delivery process, allows for its intricacy and the recently established RTB scheme, and relies on a mathematically grounded methodology.

Among other aspects, future research should explore possible improvements on the identification and harvesting of ads. Currently, our extension requires the landing page of an ad to categorize it, but we intend to use optical character recognition technique to overcome this limitation. Another strand of future work will investigate enhancements on usability. The proposed tool revisits a small fraction of the pages browsed by the user, and it proceeds by opening a new minimized window in private mode, which might be annoying to some users.

Chapter 6

MyTrackingChoices: Giving Users Fine-grained Control over Tracking

Contents

6.1 Introduction

According to PageFair 2015 report [49], 45 million Americans (16% of online users) and 77 million Europeans had installed ad blockers [1] as of the second quarter of 2015. All this globally accounts for 21.8 billion dollars worth of blocked ads in the year 2015. Since ads fuel the free content and services over the Web, this monetary loss puts the ad-based economic model of the Web under threat. The nature of this threat has become more devastating over time since, in the early days of online advertising, ad industry ignored this potentially dangerous situation in favor of immediate benefits.

Users generally block ads due to an unpleasant ad-experience, which could be a fallout of several concerns. Some users simply do not want to see ads, while others block ads because they find them too annoying (in terms of usability) or perceive them as a source of privacy and/or security issues on the Web [49, 153, 42, 48, 44, 51, 56, 136, 200]. In fact, the reasons behind ad blocking are not mutually exclusive and some users may block ads as a result of their combinations. Apparently, the general ad-experience has deteriorated over the last few years, and this has resulted into an unprecedented use of ad blockers on the Web.

[1] All tools that eventually end up blocking ads even though they are initially designed for some other purpose (privacy, transparency, etc.)

Apart from the poor ad-experience, radical design choices adopted by ad blockers is another reason why ad blocking has become a threat to the Web economy today. A scrutiny of current ad blockers further reveals that they do not take into account: 1) the economic impact of ad blocking, and 2) the social and economic benefits of non-intrusive and rational advertising. This demands better tools that cater to users' concerns, while at the same time are designed to ensure that their economic impact does not get sidelined.

In this work, we address the evoked concerns by proposing a three-dimensional approach that combines user privacy, ad blocking and its economic impact. We target users who are not against ads but block them due to privacy concerns. Our working assumption is that users consider their visits to some web pages (for example, related to religion, health) more privacy-sensitive than to others (such as news, sports) [143, 87]. Therefore, our approach consists in providing users an option to specify the categories of web pages where they do not want to be tracked and thus, do not want to receive third-party ads[2]. Unlike current ad blockers, it allows users to continue receiving ads on categories of web pages that they consider non-sensitive. In concrete terms, the proposed approach attempts to find a trade-off between privacy and economy.

To test the viability of our approach, we have developed an extension for Google Chrome called *MyTrackingChoices*[3]. While users browse the web, the extension categorizes the visited web page on the fly and, depending on the users' choices, it blocks the network connections of third-party domains present on the web page. As tracking is prevented by blocking third-party network requests, we avoid not only tracking but also third-party ads. It is worth noting that, unlike other ad blockers, our extension does not block ads served directly from the publisher on both sensitive and non-sensitive web pages.

Contributions. We make the following contributions in this chapter.

- We review existing ad blockers for their design choices and the ensuing impact on the Web economy. Our findings reveal that their approach to ad blocking is not appropriate both to users and the economy. We also show that current self-regulatory initiatives by the ad industry and various other organizations are not sufficient because users' choices are not guaranteed to be enforced.

- Building on the lessons learned, we propose a new approach that is user-centric and gives fine-grained control to users. Being user-centric ensures that user privacy and ad preferences are enforced on the user-side while fine-grained control provides users a a trade-off between privacy and economy. Our solution consists in providing users with an option to select categories of web pages that are privacy-sensitive to them and block network connections of third-party domains (and thus, ads delivered through third-party domains) present on such web pages. To test the viability of our proposed approach, we implement it as a Google Chrome extension.

- We present some preliminary results based on the pseudo-anonymous data collected from 96 users of the *MyTrackingChoices* extension. Our results regarding the impact of MyTrackingChoices on the Web economy show that the economic impact of ad blocking exerted by privacy-sensitive users can be significantly reduced. We also evaluate the usability and the performance aspects of MyTrackingChoices which confirm that it can be adopted on a wide scale.

Outline. Section 6.2 discusses the problems and perspectives associated to existing ad blockers. In Section 6.3, we present our new approach to ad blocking and its implementation as *MyTrackingChoices*. The impact of *MyTrackingChoices* on the Web economy

[2]Such ads are delivered by domains other than the one navigated by a user.

[3]`https://chrome.google.com/webstore/detail/mytrackingchoices/fmonkjimgifgcgeocdhhgbfoncmjclka?hl=fr`

is evaluated in Section 6.4 whereas Section 6.5 assesses the usability and performance of *MyTrackingChoices*. In the end, Section 6.6 measures the presence of third-party domains and Section 6.7 concludes our chapter with some directions for future work.

6.2 Related work

Due to proliferation of intrusive[4] and privacy-invasive ads, ad blockers have become very popular. They can be divided into two classes: "ad blockers" and "anti-trackers". Ad blockers are those which exists solely for blocking ads whereas anti-trackers exist for other reasons like privacy and transparency, but eventually block ads. Some examples of ad blockers are AdBlock [122], AdBlock Plus [2], etc. whereas Ghostery [11], Disconnect [8], PrivacyBadger [21] are examples of anti-trackers. In terms of functionalities, ad blockers in the first class block all ads whereas anti-trackers allow users to block a particular tracker or a particular category (related to analytics, privacy, advertising, etc.) of trackers or trackers on a per domain basis. Considering this distinction in terms of their functionalities and objectives, in the remainder of this section, we make the distinction between them.

Problem with current ad blockers and anti-trackers. Ad blockers are too radical because they take a binary approach towards ad blocking, i.e., they block all ads. Additionally, as they simply block all ads by default and do not let users configure their choices, it is evident that the developers have not considered their economic impact on the Web.

Anti-trackers give users the option to decide by which trackers or category of trackers they do not want to be tracked. However, we argue that most users are not concerned with the trackers but the other dimension of tracking, i.e., on which web pages they do not want to be tracked. In this respect, anti-trackers let users decide to block trackers on a per domain basis, i.e., users can whitelist or blacklist a specific domain. However, we argue that domain-level granularity is not the right approach for three reasons:

1. Given the huge number of domains, it is almost impossible for users to determine and pre-define all the domains where they do not want to be tracked.

2. Some domains can host web pages belonging to different categories, for example, belonging to both sensitive (health, religion, etc.) and non-sensitive (sports, science, etc.) categories [40]. Domains belonging to the "news" category, e.g., cnn.com or bbc.com, are good examples because they usually include web pages from a variety of categories (sports, economy, politics, health, travel, religion, etc.). Therefore, except some domains that have all web pages belonging to a same category, it makes more sense to block trackers based on the category of the web page and not on a per domain basis. In fact, page-granular (and not domain-granular) blocking makes more sense both from users' privacy point of view and with respect to reducing the economic damage incurred by publishers. The reason being that, for a given domain, ads would eventually be blocked only on web pages belonging to sensitive categories chosen by users and not on the whole domain.

3. Blocking trackers based on the category of web pages (instead of configuring it for each and every domain) makes it easier for users to configure as they just need to select once the categories of web pages that are sensitive to them. As the number of web page categories are limited, these pre-defined categories can be made available to the user when the tool is installed and user choices can be respected from the very beginning.

[4] Ads are considered to be intrusive if they hide content or if they pop up randomly on the screen making user browsing experience frustrating.

With the exception of PrivacyBadger [21], another issue with existing ad blockers and anti-trackers is that, in order to block trackers or ads, they rely on black-lists manually maintained by their developers or, in some cases, by user communities. The use of these lists by AdBlock Plus [2], currently the most popular among ad blocking tools, stirred controversy for accepting money from some ad companies to whitelist them [90, 28]. Furthermore, it is very cumbersome to maintain these lists as 1) new trackers and/or ad companies keep appearing in the market and 2) already existing companies keep introducing new mechanisms to deliver ads.

Self-regulatory initiatives by the ad industry are not appropriate. Without giving control to the user (i.e., providing users a tool), various self-regulatory initiatives [46, 43] from the ad industry have proposed solutions to counter some of the concerns behind ad blocking (like annoyance, usability and performance). These efforts envisage to improve the ad experience for users but none of them gives the control back to users. Moreover, these initiatives from the ad industry do not take into account the privacy concerns behind ad blocking even though various studies confirm that a non-negligible number of users block ads because of privacy concerns [49, 48, 44]. The LEAN program from IAB talks about non-invasive ads but it is not in terms of privacy [46]. "Acceptable ads manifesto" include five points to improve the ad-experience and thus, making them acceptable. However, none of these five points tackles privacy concerns [43].

Other examples of self-regulatory initiatives are "Your online choices" (from a group of European organizations) [34] and DNT (from World Wide Web Consortium) [50]. Your online choices lets users block ads tailored to their web browsing interests but users can never be sure if their choices are actually honored and if they are still being tracked. These initiatives do not suffice since users' choices are not technically enforced at the user side. Similarly, DNT allows users to notify websites and ad industry if they want to stop being tracked through third-party cookies. However, the problem of enforcement of user choices still persists.

6.3 MyTrackingChoices

As we have discussed in the Section 6.2, current ad blockers and existing self-regulatory initiatives are not appropriate to tackle the current threat to ad-based economic model of the Web. Today there is a need of better tools that give users more fine-grained control over tracking and thus, third-party ads.

6.3.1 New approach to ad blocking

We propose a different approach as a solution to the afore-described concerns. Our approach targets users who are not against ads but block them due to privacy reasons. It is based on the assumption that most people do not want to be tracked on "sensitive" websites (for example, related to religion, health), but accept to be tracked and receive ads on less sensitive ones (such as news, sport) to support the content provider.

The underlying objective is to sustain the ad-based economic model of the Web and hence, is in contrast to the design rationale of existing ad blockers. The idea is to let users choose the categories of web pages that are privacy-sensitive to them (for example, health, religion) and exclusively block the respective trackers and/or ads on those web pages. The granularity of selecting where users accept ads gives it an advantage over other ad blockers in various respects as mentioned in Section 6.2. Thanks to our approach, users can continue receiving profile-based targeted ads[5] on categories of web pages they accept

[5]As users are okay to be tracked on some categories of web pages. They still can receive useful ads targeted to those interest-categories of users.

to be tracked.

Our approach does not follow an "all-or-nothing" policy as opposed to current ad blockers. We allow users to select the categories of web pages that they consider sensitive and do not want to be tracked and receive ads on. Additionally, users can be even more selective in the sense that they can block a web page without blocking the entire category. For instance, if a particular web page hosts intrusive ads that belongs to a non-sensitive category, users can selectively penalize it while allowing ads on other web pages in the same category. This feature may encourage publishers not to include intrusive ads on their web pages.

It is worth noting that our proposed approach is not a replacement but complements other initiatives of the ad industry [46, 43]. While these initiatives aim to improve the quality of ads delivered to users, we give the onus to users to decide where they want to accept tracking and third-party ads. We emphasize that users who outrightly reject all ads are not considered. Other economic models (for example, subscription-based access to the content) must be put in place for such users [52].

MyTrackingChoices is a tool that implements our approach and is described in more details in the following section.

6.3.2 Implementation Details

MyTrackingChoices is developed as Chrome extension and is available to download from Chrome Web Store [55]. In terms of functionalities, it allows users to block trackers (and thus, third-party ads) based on pre-defined categories of web pages or on a per web page basis. If users do not agree with the categorization of a web page by MyTrackingChoices, they are allowed to change the category of that web page. Users can also view the list of third-party domains present on a web page.

MyTrackingChoices extension on Chrome currently supports web pages in English, French, Spanish and Italian. This is due to the limitation of our current "Categorizer" module (described next) but we plan to extend support for more languages in the near future.

MyTrackingChoices is composed of three main modules namely "Categorizer", "Policy module", and "Blocking module". Categorizer module is already presented in section 5.4.2. Below is the detailed description of other two modules. We note that, in the rest of the paper, the terms "web page" and "URL" are interchangeably used.

Policy module

This module is responsible for applying the blocking policies defined by users. Policies can be defined either on per category basis or on per web page basis.

To simplify the implementation, our extension allows users to define only per category blocking policies. That is, instead of specifying categories of web pages where tracking should be allowed and categories of web pages where tracking should be blocked, we just enable the latter blocking declaration.

Fig. 6.1 shows the configuration panel where users may configure the per category blocking. After having installed the extension and before the extension starts functioning, users are presented with the options panel so that they can configure their choices (by default, no categories are blocked). Configuration of blocked categories can also be performed at a later stage by clicking on "configure your tracking choices" option in the popup page of MyTrackingChoices.

Contrary to the per category policies, users are allowed to configure both block and allow per URL policies. This is to allow users to be more granular if they are occasionally not satisfied with per category blocking policies. For example, in a scenario where a user

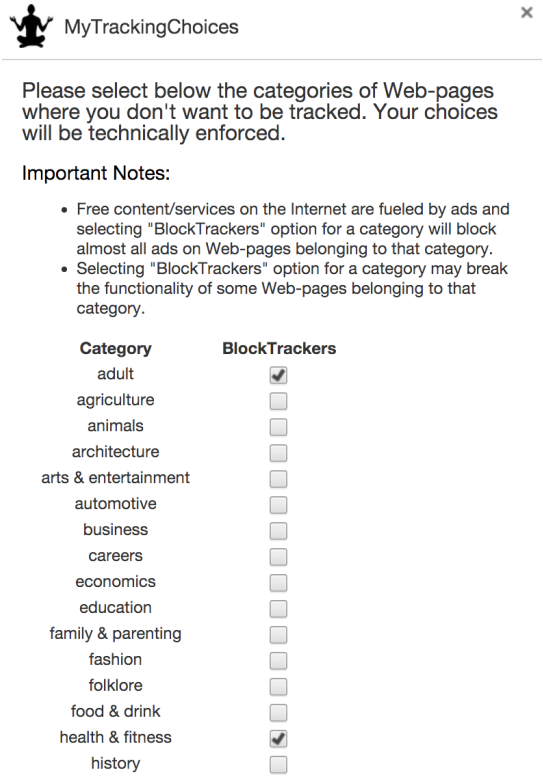


Figure 6.1: The configuration panel shown in this figure allows users to define per category blocking policies.

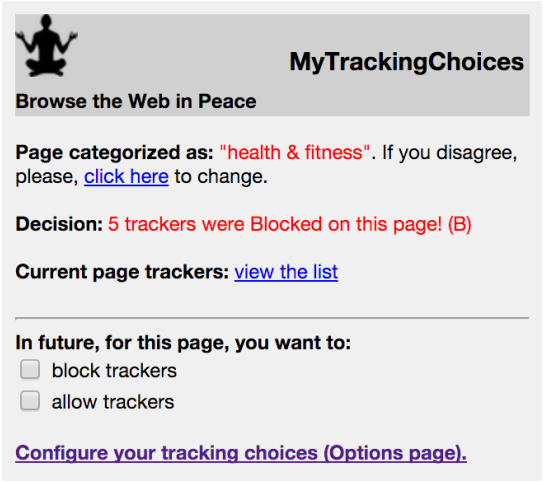


Figure 6.2: Screenshot of the popup page. It shows the category(ies) of the current visited page along with the decision about blocking or allowing the trackers. Users are also given the option to block or allow trackers the next time they visit that web page.

has blocked a category of web pages but wants to support a specific web page in that category by receiving ads. Fig. 6.2 shows how the popup page of MyTrackingChoices looks like. It provides users with options to define per URL policies.

The operation of this module is described next. When a user visits a page, the module waits for the categorizer to send the topic category of the web page browsed. Being equipped with the URL as well as the category, the module decides whether the network connections of third-party domains should be blocked or not. We emphasize here that in case of a conflict between per web page and per category based policy, per URL based policy prevails. In case a web page is categorized in more than one category, if any of the category is selected by the user to be blocked, third-party network connections on that web page are blocked.

Blocking module

This module is responsible for blocking network connections of third-party domains if a user has selected to block them. There are two main tasks. First, finding third-party domains and then, second, to block them. To find third-party domains, we just need to check if the domains of network connections match with the domain the user typed in the address bar. If it is not the same, then, such domains can be considered as third-party domains.

The next task is to block network connections to such domains. However, blocking network connections of all third-party domains may break the functionality of some web pages. This is because of the fact that some web pages download useful content from other domains (a different domain belonging to the first-party domain or a content provider or some other domain). To counter this problem, existing ad blockers keep the list of

domains that deliver ads or are tracking users. More specifically, this list of domains is also accompanied with some regular expressions that are used to pinpoint only some network connections from those domains.

MyTrackingChoices takes a different approach to find which network connections of third-parties to block so that the functionality of the web page is not broken. Instead of collecting all tracking and advertising domains (which are huge and dynamic), we build a list of domains that are essential for the functionality of a web page. In our experience, such a list is smaller and easier to maintain than the list of tracking and advertising domains. Therefore, we only keep the list of such domains [54]. We block all other domains that do not belong to this list and are classified as a "tracker" based on a heuristic described below.

We classify a third-party domain as a "tracker" if it is present on three or more different domains that a user visited in the past. This implies that the extension becomes fully functional only after a user visits a couple of web pages after installation. This heuristic is employed so that we do not need to maintain a list of third-party domains that are specific to a first-party domain to deliver content. For example, "lemonde.fr" uses "lemde.fr" domain to deliver content to their web pages. However, this domain is only specific to "lemonde.fr" and is probably used by Le Monde only for this purpose. Therefore, such domains are not classified as "trackers". As we do not need to keep the list of such domain specific third-party domains, this helps us to keep our list of useful domains reasonably small.

6.4 Economic Impact

This section assesses if our approach can contribute to decreasing the impact of ad blocking on the Web economy. This assessment is based on the data collected from users of MyTrackingChoices. In the following, we first describe our dataset and the ethical considerations regarding the data collection and use. And in the sequel, we measure the economic impact of MyTrackingChoices.

Dataset. Before the extension is put on the Chrome Web Store[6], it was distributed among friends and colleagues for beta testing. Data from these users are not collected and used in our study. The collected data is from those users of MyTrackingChoices who downloaded it directly from Chrome Web Store. Such users either found the extension on the Chrome Web Store independently or came to know about it through various publicilty channels (word-of-mouth, twitter, facebook, etc.). The dataset, used for different analyses in this chapter, includes data from January 11 to February 20, 2016.

The dataset contains 96 users[7]. These 96 users are those who browsed 20 or more web pages and configured at least once their options regarding blocking of third-party domains (either on a per category or on a per web page basis). Users, who did not configure blocking feature at all, have not been considered in this study because the installation of the extension serves no purpose in this case. The choice of minimum browsing of 20 web pages is somewhat arbitrary but the intuition behind this is not to take into account the preferences of users who did not considerably use the extension. Such users may further distort the ensuing analysis.

Based on data uploads from users to our server, we find that different users used MyTrackingChoices extension for different time intervals ranging from a minimum of around 6 hours to a maximum of 44 days. Even though it is possible for users to continue using the extension and just opt-out of data upload, we consider that they stopped using the extension when no data is received from users. There were 7.29% users (7 users out of 96

[6] `https://chrome.google.com/webstore/detail/mytrackingchoices/fmonkjimgifgcgeocdhhgbfoncmjclka?hl=fr`

[7] We consider one installation as one user. However, it is technically possible for a user to uninstall the extension and install it again. We assume that users did not do this.

users) who used our extension for less than 1 day. A vast majority of users used it for more than 1 day.

Ethical considerations regarding data collection. The collected data is considered to be pseudo-anonymized because it does not contain any information that can be directly used to re-identify users. It contains hashes and categories of web pages visited by the users along with third-party domains and URLs of iframes present on those web pages. As a part of responsible disclosure, users were informed that the extension is developed by Inria as a research project and collected data will be used to study the impact of MyTrackingChoices on the Web economy. We provided users with the option to opt-out of data collection as well as delete previously uploaded data to our servers. Furthermore, we assured users not to make an attempt to re-identify them and also, not to share the data with any outside party. MyTrackingChoices privacy policy[8] clearly specifies all the details regarding our data collection, storage, processing and sharing.

6.4.1 Analysis

The analysis is done under the reasonable assumption that users know how to properly use the extension. To ensure proper functioning of the extension, we had put a clear notice[9] on our project web page[10] as well as extension download page [55] on Google Chrome Store to uninstall other ad blockers or disable blocking functionality in other anti-trackers. Indeed, such extensions could intervene in the functionality of MyTrackingChoices and therefore, user choices may not be completely respected.

In the following, we present results cumulated over all the users of *MyTrackingChoices* since our goal is to measure the global effect on the Web economy. As we are interested in different aspects related to different categories of web pages, most of our results do not focus on users but categories of web pages.

To measure the impact on economy, we first start by looking at number of users who actually exercised fine-granular choices. We find that 69.98% users accept the presence of third-party domains and thus, tracking and hosting ads in some categories of web pages that they do not consider much sensitive. On an average, a user blocked 15.28 categories (almost half) out of a total of 32 categories. The median of the number of blocked categories is 11. These bare statistics suggest that the impact on the Web economy due to ad blocking can be reduced if users are provided with such fine-grained choices. However, we need to keep in mind that such a solution is for users who block ads because of privacy reasons, who are not against ads and do not want to be tracked on some categories of web pages that they consider sensitive.

We also note that 30.02% (29 users out of a total of 96) users selected all categories to block network connections of third-party domains. These are probably the users that are either against tracking altogether or probably the ones who do not want to see ads at all. In fact, blocking trackers on all visited web pages will block almost all ads as most ads are delivered today through third-party domains present on a web page. As these users simply do not want to exercise fine-granular choices, providing them such an option will not be help survive the ad-based Web economy.

Next, we are interested in knowing categories of web pages that are most or least blocked by users. This should enable us to know the categories that are the most sensitive to users (where most users do not want to be tracked) and categories that users care the least with respect to tracking. Equipped with this information, in future, we can propose

[8] Available at `https://myrealonlinechoices.inrialpes.fr/privacy_policy.html`

[9] Here is the excerpt from our web page: "Please uninstall other Adblockers or disable blocking functionality in other anti-tracking extensions like Ghostery, DisConnect, PrivacyBadger etc. before installing MyTrackingChoices. Otherwise, your choices would not be completely respected."

[10] `https://myrealonlinechoices.inrialpes.fr/`

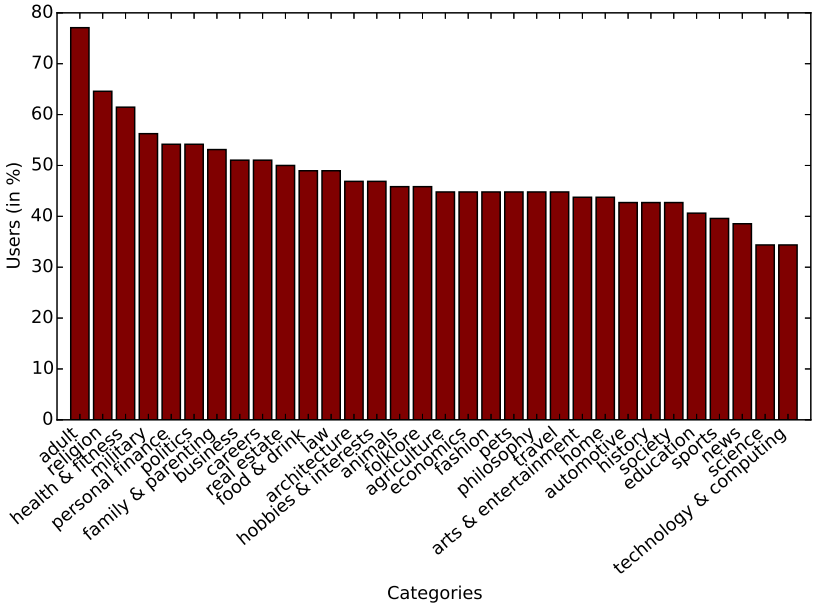


Figure 6.3: Percentage of total users blocking a category. Total number of users considered in this study are 96.

to users the categories of web pages to block by default and let user tweak them if they are not satisfied.

Fig. 6.3 presents the categories blocked by users in the order of the most blocked to the least blocked. We note here that the three most blocked categories are "adult", "religion" and "health & fitness". These categories are, in fact, also considered as *sensitive* by European Data Protection Law [40]. The three least blocked categories are "technology & computing", "science" and "news". This result confirms that users are more concerned about sensitive categories [40]. Therefore, it actually makes sense to provide them with such an option, i.e., to block trackers based on the categories of web pages.

As we now know the most and the least blocked categories by users, it is interesting to study the browsing habits of users (category of web pages they browse the most/least) to get more insights about the impact on the web economy. The total number of browsed web pages by these 96 users are 86,922, out of which 12,543 are distinct. Fig. 6.4 presents the distribution of total and distinct visited pages per category. This also shows the distribution of blocked versus allowed pages in each category. In order to detect any bias in the distribution of browsed web pages (owing to small number of users or type of users), we check it with other studies and the pattern seems to resemble well [118, 137].

We find that the top-5 most browsed categories by our extension users are "society", "technology & computing", "arts & entertainment", "science" and "hobbies & interests". Here it is worth noting that web pages browsed in these 5 categories constitute for roughly 65% of the total browsing activity. However, none of these top-5 most browsed categories (Fig. 6.4) are in the top-5 of the most blocked (Fig. 6.3). This explains why there are more allowed than blocked web pages in the top-5 categories of Fig. 6.4.

There are more blocked than allowed web pages, in the Fig. 6.4, for the categories that are the most blocked in Fig. 6.3. As the most blocked categories are least browsed and the most browsed categories are the least blocked, the effect on the economy is less damaging. In total, only 33.19% of browsed web pages were blocked. This shows that the number of blocked web pages are far too less than the ones where trackers (and thus, third-party ads) are allowed.

Until now, we studied the effect on the economy due to fine-grained blocking options

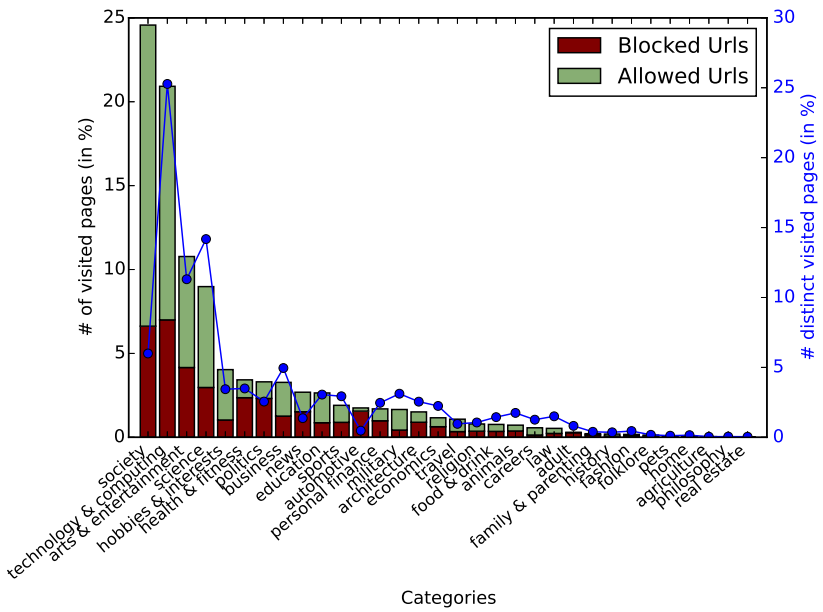


Figure 6.4: Distribution of blocked and allowed visited pages by category. Right side y-axis presents distribution of distinct visited web pages by category. The total number of browsed web pages are 86,922 whereas the total number of distinct web pages browsed are 12,543.

on per web page basis. This already indicates that MyTrackingChoices can reduce the impact of ad blockers on the Web economy. However, as we also collect the source URLs of all iframes included in the web pages browsed by users, this allows us to measure how many iframe-based ads actually got blocked. As iframes are generally used to deliver ads from third-party domains directly into a web page, we can make an accurate estimate of the impact of *MyTrackingChoices* on the Web economy. It is worth noting that this study excludes non-iframe based ads as well as those delivered directly by the publisher. However, as per our experience, percentage of such ads is very low.

In total, we find 958 different domains delivering iframes over all 96 users. However, since all iframes do not necessarily contain ads, we first filter those iframes that are used to deliver ads. This filtering is done based on the list of advertising domains from Mozilla Focus project and their partner Disconnect [53]. After this filtering, we find a total of 115 different ad domains and a total of 27,861 iframes or ads served by these domains. Table D.1, in the appendix D of this thesis, contains top 40 ad domains that delivered ads through iframes.

Fig. 6.5 presents per category distribution of ads. For each category, we also have the fraction of ads that are allowed or blocked due to blocking policies specified by users. The categories of web pages which host most of the ads are “arts & entertainment”, “technology & computing” and “news”. As expected, we find that the percentage of ads the most blocked are in categories that are most blocked by users, for example, “adult”, “health & fitness”, “religion”, “politics”, “personal finance”. However, as web pages belonging to these categories are comparatively less visited by users as well as such web pages have comparatively lesser number of ads, the total number of ads blocked by all users of *MyTrackingChoices* is small. In total, only 23.8% of all ads are blocked.

Conclusion. Assuming that our collected data is representative of users blocking ads due to privacy reasons, we believe that the economic impact of ad blocking exerted by privacy-aware users can be significantly reduced if we provide users with such a fine-grained control. As the loss due to ad blocking was 21.8 billion dollars in the year 2015 according

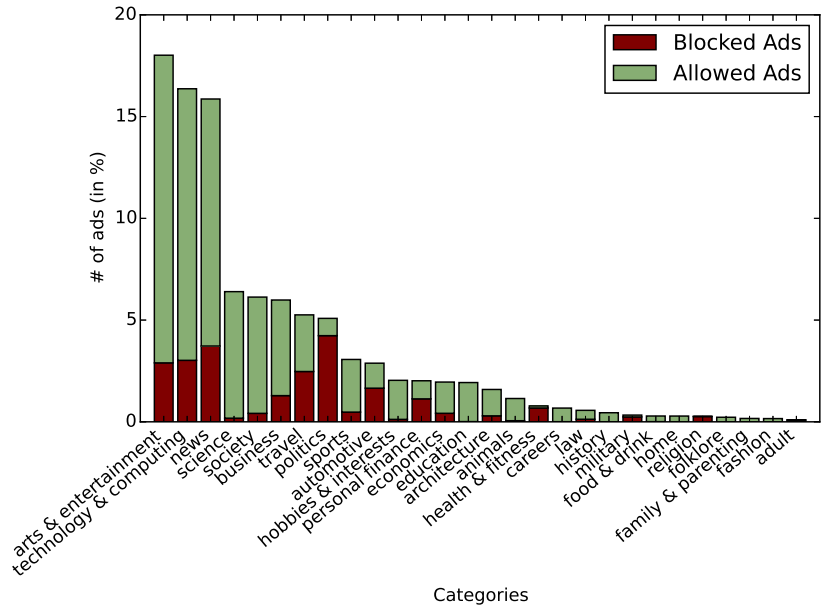


Figure 6.5: Distribution of ads by category and on the fact if they are blocked or allowed. Total number of ads are 27,861.

to PageFair 2015 report [49], this could have been reduced to 14.7 billion dollars with the use of *MyTrackingChoices*. We estimate this reduction in the economic loss based on the facts that only 23.8% ads are blocked by *MyTrackingChoices* users and 50% of users block ads due to privacy concerns [49].

6.5 Usability and Performance Evaluation

This section assesses the usability and performance of MyTrackingChoices. We report upon the pseudo-anonymous data collected from users of our extension.

6.5.1 Usability

As usability is an essential factor for a wide scale adoption of a tool, we evaluate two of its important aspects. Before going into further details, we reiterate that current anti-trackers allow users to block ads based on a per URL basis whereas MyTrackingChoices lets users specify their blocking policies based on categories of web pages. Bearing in mind this distinction, our first evaluation criterion considers the frequency with which users need to change the two blocking policies. Under this criterion, we compare the blocking policies used by current anti-trackers and MyTrackingChoices. It is evident that category-based blocking should outperform URL-based blocking. However, for the sake of completeness and accuracy, we measure the performance of MyTrackingChoices in this regard.

We note that in certain circumstances, category-based blocking can be too coarse. For example, a user may not want to block a specific web page in a blocked category or vice versa. To this end, MyTrackingChoices, offers an additional possibility to be even more granular and block or allow on per web page basis. Our second criterion captures the usage of this feature. This criterion is evaluated by counting the number of users who exercise such a granular feature and at how many web pages this feature is used.

In the following, we measure the afore-described criteria on our dataset.

First, we study how users' tracking choices evolve over time. Specifically, we want to know if users are satisfied by just configuring once their tracking choices when they install

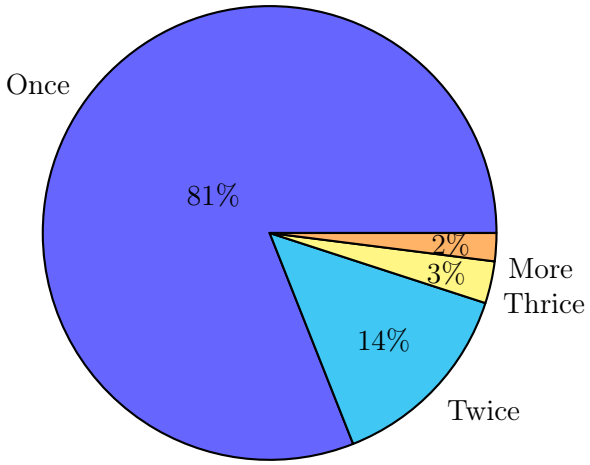


Figure 6.6: Percentage of users modifying the "category-wise blocking of trackers setting" different number of times.

the extension or they change their tracking choices post-installation. Fig. 6.6 presents the percentage of users who change their tracking choices. We note here that 81% users configure their choices once and then, do not feel the need to change it. This signifies that blocking based on categories is stable over time and this makes it highly usable. Additionally, we highlight that almost all (16 out of 18) remaining users added one or more categories, while only 2 users removed an already blocked category.

Next, we measure how users exercised per URL blocking option. We find that 52 users out of a total of 96 users, i.e., 54.17% users, used this functionality. This clearly suggests that such an option is useful to have. In total, web pages where URL-based blocking feature is used, constitute for 1.01% (127 web pages out of 12,543) of total web pages browsed by these 52 users. This is not a considerable percentage but we need to keep in mind that all such users also exercised their choice regarding blocking trackers based on the categories of web pages. They usually need to exercise per web page blocking in case they are not satisfied with their per category blocking decision.

Fig. 6.7 presents the distribution of per URL granular policies exercised by these 52 users over different categories of web pages. It is worth noting that the top-5 categories for which per URL policies are defined, none of them belongs to the top-5 categories blocked by users (Fig. 6.3). This means that the web pages which contribute the most to the per URL policies are those that belong to non-sensitive categories. Also, we note that 37.42% of per URL policies are defined to allow trackers on different web pages where tracking is blocked as per their category. This leads us to the conclusion that some users are willing to accept ads on web pages belonging to non-sensitive category and indeed want to support publishers. However, at the same time, we also notice that trackers/ads are blocked on a per URL basis for web pages that are not blocked based on their categories. This implies that users do block trackers and thus, eventually ads on certain web pages even if they did not consider their categories sensitive. This user behavior suggests that these web pages may contain intrusive ads and therefore, users opted for blocking of third-party domains.

6.5.2 Performance

We measure the performance of *MyTrackingChoices* in various respects. First, we measure the performance of our extension in terms of correctness of categorization algorithm and proper functioning of the web pages. We find that incorrect categorization is reported for only 18 web pages (out of a total of 12,543 distinct web pages browsed by all the 96 users) by 12 users. This shows that users are mostly satisfied with the categorization of web pages

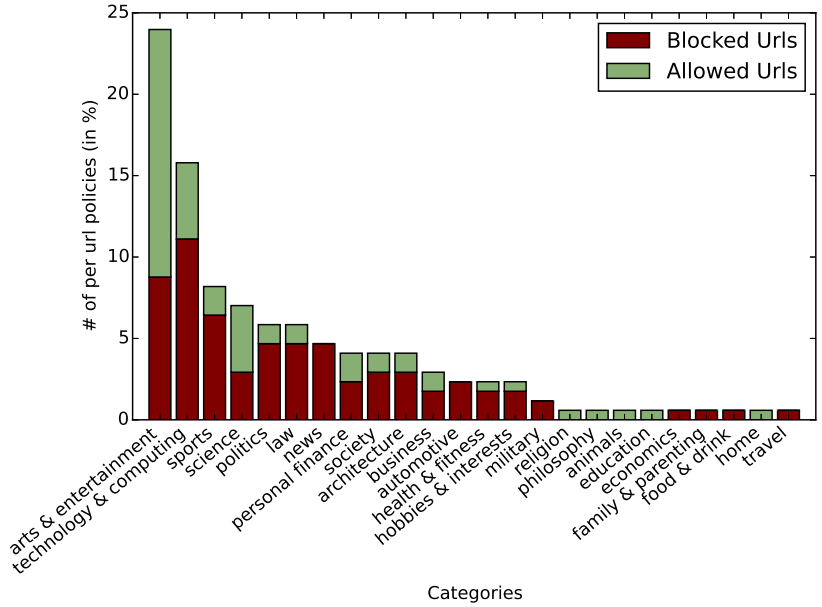


Figure 6.7: Distribution of per URL defined policies by category.

by our extension. Also, seven users reported 23 web pages whose functionality broke due to MyTrackingChoices. Most of the times, it was because of blocking of content-provider domains and we corrected the problem by updating the list of allowed domains.

Next, we assess our extension in terms of average page-loading time, maximum memory usage and maximum CPU usage, and compare these results with some of the most prominent ad-blockers and anti-trackers. This is necessary because our extension provides certain privacy guarantees in terms of user tracking but it comes at the cost of processing and computational overhead.

Fig. 6.8 presents the benchmark analysis of this comparison. The results of this benchmark analysis have been obtained after ten consecutive visits to these three popular web pages: cnn.com, lemonde.fr and nytimes.com. The main conclusion that can be drawn from this figure is that our extension, despite its higher complexity, outperformed most of the tools evaluated in this study. In terms of page loading time, MyTrackingChoices is close to Ghostery and uBlock Origin, the two tools which led this aspect. As for memory usage, our extension required between 37 and 39 MB, ranking third after Ghostery and uBlock Origin in lemonde.fr and nytimes.com, and second after Ghostery in cnn.com. Finally, with just 2% of processing usage, our extension outperformed all tested tools in terms of CPU consumption. In a nutshell, the performance of MyTrackingChoices was above the average for those three web pages, and this was despite the inclusion of more sophisticated and advanced privacy features.

6.6 Presence of Trackers

In this section, we study the prevalence of trackers on the Web. We first discuss how this study is conducted and then, present the measurement results.

If a third-party domain is present on a web page, we assume that it is tracking users. Moreover, a third-party domain is classified as “tracker” only if it is found to be tracking users on three or more web pages (as described in Section 6.2). The consequence is that first two presence of a third-party domain does not suffice for the domain to be classified as a “tracker”. In addition, we exclude from our study web pages that are blocked by

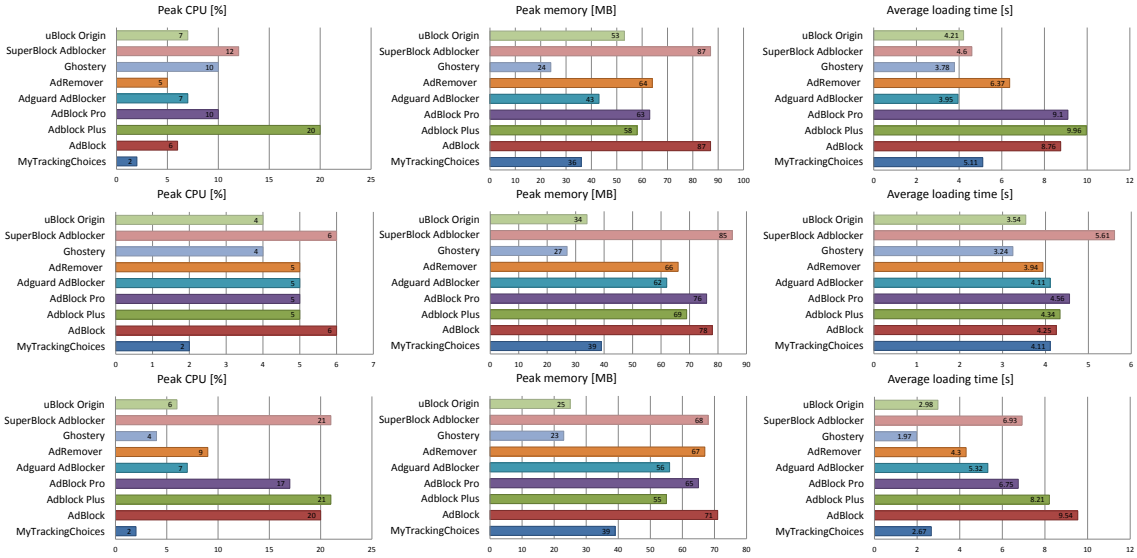


Figure 6.8: The first row shows the peak CPU usage (in %), peak memory usage (in MegaBytes) and average loading time (in seconds) for home page of CNN website (cnn.com). The second and third rows show the same information for home pages of Le Monde (lemonde.fr) and New York Times (nytimes.com) respectively.

users. This is because if the initial request to a tracker is blocked, none of the consequent requests, that would have been made otherwise, can be captured.

Users in our dataset were tracked 239,420 times by a total of 588 different trackers. The average number of trackers, over all users, per web page is 3.2 with a standard deviation of 5.1. However, the maximum number of trackers present on a web page can be as high as 98 in some cases whereas the minimum is zero.

Fig. 6.9 presents the distribution of trackers over different categories. From this figure, we note that web pages in the "news" category contain the maximum number of trackers on average. However, if we contrast it with number of ads received on web pages in the "news" category (Cf. Fig. 6.5), we find that the web pages in the "news" category did not receive the maximum number of ads. This suggests that web pages in the "news" category include many additional trackers other than those related to ads. In fact, the average number of trackers in the "news" category is 14.28 with a standard deviation of 10.37. This is much higher than the global average of 3.2 trackers per page.

From the same figure, it is interesting to note that web pages in the "technology & computing" have the most number of different trackers, close to 500, even though the average number of trackers per web page is quite small. In fact, web pages categorized as "news" are tracked by 284 different trackers even though the average number of trackers per page is the highest. This result suggests that the trackers are most spread in web pages belonging to "technology & computing" category.

We refer interested readers to Table D.2 in the appendix D of the thesis which shows the list of frequent trackers overall and in some of the sensitive category web pages.

6.7 Conclusion and Future Work

Considering the poor ad-experience on the part of users, ad blockers and anti-trackers have become an indispensable tool. They provide absolute privacy to users at the cost of an adverse impact on the Web economy. In this chapter, we proposed a new approach that provides a balance between user privacy and the Web economy. The proposed approach is user-centric and allows users to exercise fine-grained options. It allows users to choose the

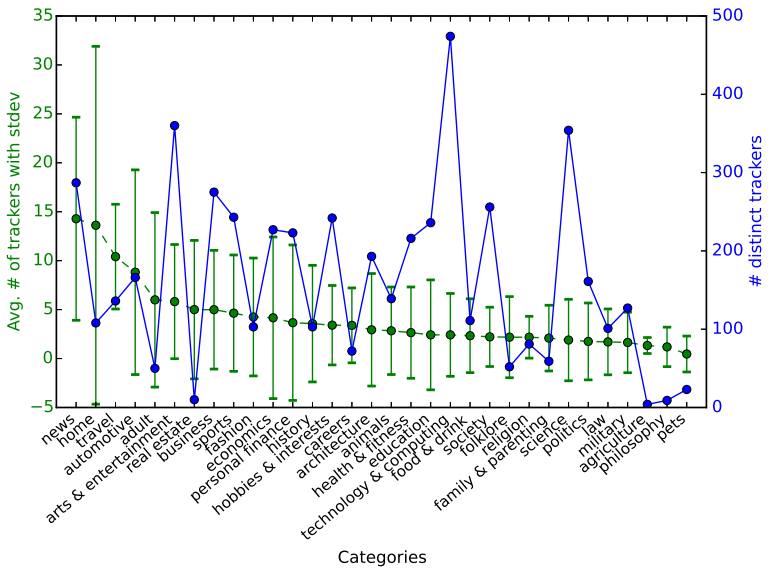


Figure 6.9: Distribution of trackers by category. Left-side y-axis (in green) represents average number of trackers with standard deviation whereas right-side y-axis represents number of distinct trackers.

categories of web pages that are privacy-sensitive to them and block trackers on such web pages only. As a result, it helps users to control their profile collected by different third-parties, e.g., advertising companies, data brokers, insurance companies, etc. We believe that our proposed approach can provide the desired level of privacy while simultaneously supporting the ad-based economic model of the Web. Overall, it can also help improving the user ad-experience and the quality of ads.

As a future work, we aim to extend our tool by proposing users an option to block annoying ads irrespective of where they appear. Another line of work consists in letting user specify the maximum number of ads they are willing to receive on a web page belonging to a non-sensitive category web page. We believe that the categorization of webpages is the most crucial part in this approach and therefore, the future work should consist in possible improvements in categorization algorithm. Machine learning techniques can be very helpful for this task and therefore, they should be explored to build a better categorization algorithm. We plan to block trackers by default on sensitive category web pages but users would have the option to tweak the default settings if they do not agree. Another direction of future work could be automatically learning the categories of web pages a user wants to block based on the web pages she blocks. A recent survey pointed out that users want to block the trackers on web pages that they visit more frequently, i.e., the frequency of the visits could be an important parameter for a user to block or not the trackers on a webpage. The future versions of this work should try out this approach too.

Chapter 7

Discussion, Conclusion, and Future Directions

Global summary. Apart from the problem of mass surveillance by government agencies, users are constantly being tracked and profiled by marketers and other third-party (intermediate) companies. The motivation for these companies is to deliver targeted ads to users and therefore, have better ad revenues. As smartphones and the Web today are the two most important technologies to access the Internet, this thesis encompassed the two and addressed the privacy problems by acting both at the origin and the end-user side. In case of smartphones, privacy problems are tackled at the origin by revealing some bad privacy practices employed by actors such as app developers, OS providers, and A&A companies. In case of the Web, by providing users with tools to have more transparency and fine-grained control, we tackled the privacy problems at the end-user side.

More specifically, on smartphones, we designed and implemented a simple yet efficient dynamic analysis technique, named *MobileAppScrutinator*, to detect, measure and compare privacy leaks on Android and iOS. Using our tool, we compared the ongoing tracking and private data leaks on Android and iOS. We also studied and revealed the privacy implications of data we share and services we use. Finally, in case of the Web, we provided users with tools to have more transparency and control over online tracking and advertising. In terms of transparency, these tools allow users to know what profile trackers and advertisers know about them and why an ad is delivered to them. In terms of control, users are given the option to exercise fine-grained control like choosing the categories of web pages where they do not want to be tracked and the kinds of ads they do not want to receive. Providing users with fine-grained control allows users to take social and economic benefits of online advertising while technically enforcing their privacy preferences. In fact, providing such control is essential to help sustain the ad-based economic model of the Web. Our work had direct impact towards enhancing user privacy both in case of smartphones and the Web.

Discussion, Conclusion, and Future Directions. To draw our conclusion and lay out directions for future work, it is important to understand the ad-supported Internet ecosystem. Basically we need to understand why online advertising exists.

On the Internet, users typically do not pay directly or pay very less for services they use, e.g., when they download apps or surf on the web. In fact, for most part of their revenues, the service providers rely on advertising. They include code from advertising and analytics (A&A) companies in their services (apps or websites). These A&A companies collect information about the users and create user profiles. Marketers then use these profiles to select the most suitable user for their ad to get displayed. As shown in Figure 7.1, this is a relatively virtuous circle if all actors work in harmony as everyone gets benefits: users can benefit from quality services for free, the service providers can get revenues for their

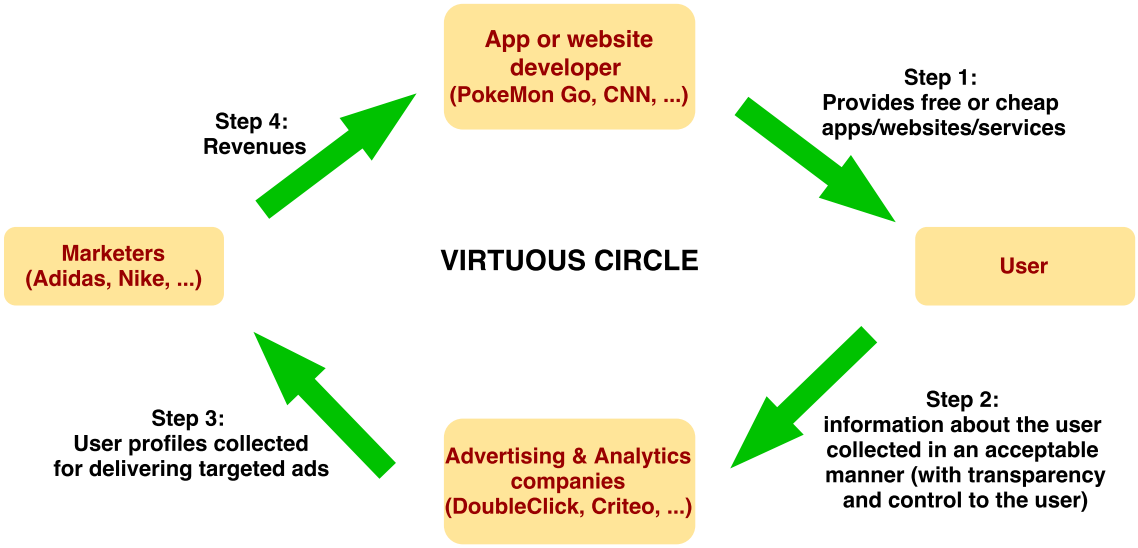


Figure 7.1: The ad-supported Interent ecosystem makes a virtuous circle if all actors work in harmony. Users get either free or cheap services (websites, apps) on the Internet. App/website developers are paid by marketers. Marketers find appropriate users for their ads to get displayed.

work, and the marketers can find users that are suitable for their ads. In conclusion, this ad-supported Internet ecosystem is not inherently bad.

However, the virtuous circle in Figure 7.1 has problems today. First, the number of A&A companies have considerably increased and made the whole system very complex. Today it is not clear for most users which A&A company knows what information about them. In addition, some actors are massively collecting data without any respect to the privacy of the users. This data collection is sometimes disproportional to the service they provide. Users are completely oblivious to what is going on; for them, the data collection is happening without their knowledge. Lastly, it is the lack of control for the users that makes the situation even worse.

With all these problems today in the supposed to be or could be virtuous circle, it is difficult to call it a virtuous circle. To make this circle really virtuous, various entities need to play a key role. Data protection agencies or regulators should define rules about data collection, processing and storage and then, check the practices of various stakeholders (A&A companies, app/website developers and OS/Browser providers) by employing regular control mechanisms. Regulators should check if the privacy policies of apps or websites matches with their actions. The accountability of each stakeholder should be fixed. In particular, service providers and A&A should be held accountable for their actions. Operating system and browser providers have the responsibility in terms of putting limits on personal data collection by A&A companies and service providers. They should provide users with enough tools to have transparency over the hidden data collection processes and then, finally, provide users with tools to have control over their data. Figure 7.2 presents where the ecosystem should tend to and fixes the responsibilities of various actors to help the smooth functioning of this virtuous circle.

We draw the following conclusion and future directions of work. We find that the current situation regarding privacy on the Internet is far from perfect: smartphone apps on both Android and iOS are leaking users' private data, the permission system on Android is not completely effective, unprotected data such as list of installed apps on Android poses privacy threats to users, and there is a lack of transparency and control for the end user. We also find that OS providers and app developers sometimes misbehave unconsciously

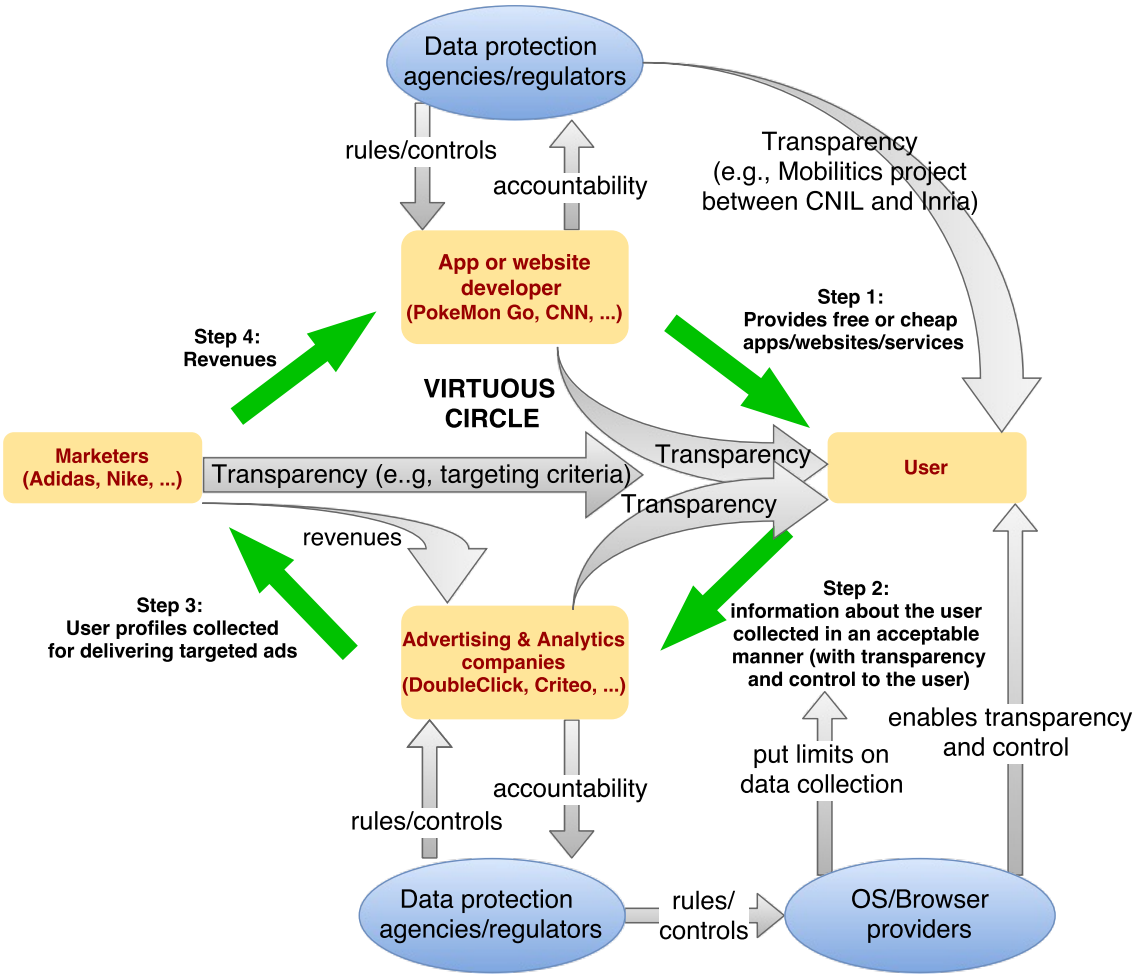


Figure 7.2: A high-level overview of where the ecosystem should tend to. Without transparency, accountability, and fine-grained user control, it is hard to make this circle virtuous.

and they are not completely aware of their actions. By revealing their mistakes, we have encountered many cases where they addressed the privacy problems themselves. They were simply not aware of privacy consequences of what they were doing. We have also seen many cases where regulators played an important role to address the privacy problems and therefore we, as a research community, should work hand in hand with regulators. Overall, research community should continue their efforts at different levels (as is done in this thesis) to enhance privacy on the Internet.

In case of smartphones, we note that the situation is different depending on the OS. Apple recently allowed iOS users to filter content. The provided APIs from Apple to block content can be used to filter trackers/ads inside apps. As soon as the API is released, plenty of ad blockers were made available on the official app store of Apple. Currently, these ad blockers are among the most popular apps on iOS AppStore. However, there is no such feature on Android.

In terms of privacy dashboard or control panel, iOS offers its users a privacy dashboard where users can authorize or revoke access to private data to an app at any point of time. In a similar attempt, Android 4.3 also introduced a dedicated control panel, called App Ops. This control panel allowed users to authorise or block access to private data at any point of time irrespective of the permissions required by the app. Users could also block the network traffic of an app. However, it was later removed from Android. This clearly shows Google's unwillingness to provide users with such a tool as the company's most of

the revenue relies on online advertising.

To track smartphone users across apps, an identifier, unique to the device, is needed as opposed to the web browsers. Therefore, the ability of A&A companies to track smartphone users usually depends on the availability of such identifiers on an operating system. Today we see that iOS does not provide any stable unique identifier to the apps. The only identifier available to apps is the advertising identifier generated randomly and that users can reset whenever they want. On the contrary, on Android, the situation is different: a variety of identifiers (hardware-tied, SIM-tied, OS-tied, or the ones that are specifically designed for tracking the device for advertising purposes) are available to be accessed. On Android, these identifiers can either be accessed by apps freely or they require a permission. However, these required permissions do not explicitely tell users about the availability of these identifiers.

In terms of future directions of work, the first priority should be to precisely understand the privacy implications of today's rich, multimodal data and complex services. This is a task that mainly researchers are capable of and are supposed to do because other stakeholders (such as regulators, privacy advocates, law people) may not be trained. Furthermore, this task is of utmost importance because it is otherwise not possible for other stakeholders to understand hidden privacy implications.

Secondly, we must work with other stakeholders towards bringing transparency because it is otherwise impossible for users to exercise proper control over their personal information even if they have the right tools. Answers to following questions should be clear to users: which data about them is being collected by a particular system/service and why, what happens to the data once it is collected, i.e., how it is stored and processed?

The third task, priority-wise, is to provide tools that are viable in the long run. This is the last step because end-users can only exercise control if they are well-informed, i.e., the systems are transparent about data collection and they do understand well the privacy implications. The tools are to give control back to the user and should be designed in such a way that they are usable and make sense for normal users (and not only to experts). Moreover, the designed tools should be as fine-grained as possible so that users can exercise their choices in a flexible manner.

Appendix A

Measuring Private Data Leaks using MobileAppScrutinator

Table A.1: Unique System Identifiers transmitted by 140 Android Apps tested.

		Server	AndroidID		PhoneNo	IMEI		SerialNo	IMSI	WiFi MAC
			Modified	Unmodified		Modified	Unmodified			
Third-parties	Clear	amazonaws.com	✓	✓		✓				✓
		ad-x.co.uk		✓			✓			✓
		mobilecore.com		✓			✓			✓
		kochava.com		✓			✓			
		apsalar.com	✓	✓						
		mdotm.com		✓			✓			
		adtilt.com		✓			✓			
		estat.com					✓	✓		
		sophiacom.fr		✓						
		appnext.com		✓						
		flurry.com		✓						
		socialquantum.ru								✓
		sitestat.com						✓		
		pureagency.com					✓			
		smartadserver.com		✓						
		xiti.com		✓						
		playhaven.com		✓						
		yoz.io		✓						
		seattleclouds.com		✓						
		ad-market.mobi		✓						
	SSL	tapjoyads.com		✓		✓	✓	✓		✓
		airpush.com	✓	✓	✓	✓				
		revmob.com		✓			✓	✓		
		appwiz.com		✓	✓		✓			
		amazon.com		✓				✓		
		adcolony.com	✓			✓				
		fiksu.com		✓			✓			
		crittercism.com		✓			✓			
		googleapis.com			✓				✓	
		appsflyer.com					✓			✓
		dataviz.com					✓			
		mobileapptracking.com		✓						
First-parties	Clear	mobage.com		✓			✓			
		ijinshan.com					✓			
		blitzer.de					✓			
		eurosport.com							✓	
		cdiscount.com							✓	
	SSL	google.com		✓			✓	✓		
		badoo.com		✓			✓		✓	
		dropbox.com		✓				✓		
		klm.com		✓						
		airfrance.com		✓						
		airbnb.com		✓						
		groupon.com		✓						
		adobe.com							✓	
Unidentified	Clear	72.21.194.112		✓				✓		
		dxsvr.com					✓		✓	
		69.28.52.39		✓			✓			
		198.61.246.5		✓			✓			
		183.61.112.40		✓			✓			
		61.145.124.113		✓			✓			
		74.217.75.7		✓						
		183.61.112.40							✓	
		linode.com				✓				
		93.184.219.20	✓							
		107.6.111.137	✓							
		startappexchange.com		✓						
		91.103.140.6		✓						
		209.177.95.171		✓						
		ati-host.net	✓							
		adkmob.com		✓						
	SSL	fastly.net		✓						
		canal-off.sbw-paris.com					✓			

Table A.2: Unique System Identifiers transmitted by 140 iOS Apps tested.

		Server	AdIdentifier	UDID	DeviceName	WiFi MAC		Pasteboard IDs
						Modified	Unmodified	
Third-parties	Clear	clara.net	✓				✓	✓
		appads.com	✓					
		amazonaws.com						✓
		1e100.net						✓
		adcolony.com	✓					
		facebook.com						✓
		your-server.de						✓
		sophiacom.fr	✓					
		smartadserver.com	✓					
		mopub.com	✓					
		sofialys.net						✓
		visuamobile.com						✓
		swelen.com	✓					
		adtilt.com	✓					
		nanigans.com	✓					
		tapjoyads.com		✓				
		greystripe.com		✓				
		mdotm.com		✓				
		sofialys.net						✓
		visuamobile.com						✓
		admob.com		✓				
		ad-inside.com		✓				
		xiti.com						✓
		2o7.net						✓
		jumptap.com	✓					
	SSL	tapjoyads.com	✓				✓	
		trademob.net	✓				✓	
		adjust.io	✓					
		boxcar.io			✓			
		flurry.com				✓		
		tapjoy.com	✓					
		mobile-adbox.com	✓					
		fiksu.com	✓					✓
		tapad.com	✓					
		testflightapp.com	✓					
		adtilt.com	✓					
		nanigans.com	✓					
		ad-x.co.uk	✓					
		crittercism.com			✓			
		facebook.com	✓					
		newrelic.com						✓
		adzcore.com	✓					
First-parties	Clear	gameloft.com	✓				✓	✓
		spotify.com						✓
		disneyis.com						✓
		mobiata.com		✓				
	SSL	gameloft.com	✓				✓	
		paypal.com	✓		✓			✓
		booking.com	✓					
		eamobile.com	✓					
		expedia.com		✓				
Unidentified	Clear	amazonaws.com	✓	✓			✓	
		igstudios.in						✓
		69.71.216.204			✓			
		198.105.199.145			✓			
		akamaitechnologies.com						✓
		softlayer.com						✓
		cloud-ips.com						✓
		mydas.mobi		✓				
		mkhoj.com		✓				
		74.217.75.7/8				✓		

Table A.3: Detection of PII leakage using TaintDroid 4.3 (Same applications are tested to compare the results with MobileAppScrutinator platform).

		Server	IMEI	Browser History	Address Book	SMS
Third-parties	Clear	hinet.net	✓			
		enovance.net	✓			
		aol.com	✓			
		kimsufi.com	✓			
		typhone.net	✓			
		betacie.net	✓			
		1e100.net	✓			
		dolphin-server.co.uk	✓			
		linode.com	✓			
		amazon.com	✓			
		ati-host.net	✓			
	SSL	amazonaws.com	✓		✓	
		1e100.net			✓	
		skyhookwireless.com	✓			
		akamaitechnologies.com	✓			
First-parties	Clear	teamviewer.com	✓			
		badoo.com	✓			
		shazamteam.net	✓			
	SSL	svcs.paypal.com	✓			
		amazon.com	✓			
Unidentified	Clear	162.13.174.5	✓			
		69.28.52.38	✓			
		195.154.141.2	✓			
		188.165.90.225	✓			
		91.103.140.225	✓			
		61.145.124.113	✓			
		69.28.52.36				✓
		183.61.112.40				✓
		91.213.146.11	✓			
	SSL	31.222.69.213	✓			
		212.31.79.7	✓			
		92.52.84.202	✓			
		69.194.39.80	✓			
		72.26.211.237	✓			
		192.225.158.1	✓			
		54.256.81.235	✓			
		67.222.111.117		✓		

Table A.4: Servers where IdentifierForVendor is communicated in 140 iOS Apps tested.

Third-parties		**First-parties**	
Clear	**SSL**	**Clear**	**SSL**
mobileroadie.com, clara.net, appads.com, adcolony.com, sophiacom.fr, 7mobile7.com, sitestat.com, mediatemple.net	tapjoyads.com, tapjoy.com, adzcore.com, fiksu.com, crittercism.com, ad-x.co.uk	eurosport.com, gameloft.com	eamobile.com, dailymotion.com, foursquare.com, google.com, googleapis.com, paypal.com

Table A.5: User PII transmitted by a total of 140 Android Apps tested.

		Server	Accounts	Contacts	Location	Operator Name	SIM Network code	WiFi Scan/Config
Third-parties	Clear	seventynine.mobi			✓	✓		
		kiip.me			✓	✓		
		google.com			✓		✓	
		3g.cn			✓			
		doubleclick.net					✓	
		goforandroid.com					✓	
		adtilt.com				✓		
		2o7.net				✓		
		nexage.com					✓	
		ad-market.mobi					✓	
		mopub.com			✓			
		mydas.mobi			✓			
		startappexchange.com					✓	
	SSL	airpush.com			✓	✓		
		appwiz.com			✓		✓	
		agoop.net			✓		✓	
		tapjoyads.com				✓	✓	
		crittercism.com				✓		
		inmobi.com						✓
		appsflyer.com				✓		
		googleapis.com	✓					
First-parties	Clear	betomorrow.com			✓	✓	✓	
		avast.com	✓			✓		
		google.com	✓	✓			✓	
	SSL	badoo.com			✓	✓	✓	✓
		checkmin.com			✓			
		groupon.de				✓		
		m6replay.fr				✓		
Unidentified	Clear	91.103.140.193	✓			✓	✓	
		adkmob.com					✓	
		dsxvr.com					✓	
		amazonaws.com				✓		
		183.61.112.40					✓	

Table A.6: User PII transmitted by a total of 140 iOS Apps tested.

		Server	Accounts	AddressBook	Device Name	Location	SIM Network Name	SIM Number
Third-parties	Clear	clara.net					✓	
		amazonaws.com					✓	
		bkt.mobi				✓		
	SSL	capptain.com				✓	✓	
		fring.com						✓
		crittercism.com			✓			
		boxcar.io			✓			
		testflightapp.com					✓	
First-parties	Clear	mobilevoip.com		✓				
		groupon.de					✓	
		sncf.com				✓		
	SSL	groupon.de				✓	✓	
		ebay.com						✓
		foursquare.com				✓		
		paypal.com			✓			
		twitter.com	✓					

Table A.7: Different Pasteboard Names, Types and Values created by 140 iOS Apps tested.

Pasteboard Names	Pasteboard Types	Pasteboard Values
fb_app_attribution, org.OpenUDID.slot.0 to 99, com.hasoffers.matsdkref, com.ad4screen.bma4s.dLOG, com.ad4screen.bma4savedata124780, com.flurry.pasteboard, com.fiksu.288429040-store, com.fiksu.pb-0 to 19, org.secureudid-0 to 99, com.ebay.identity, com.paypal.dyson.linker_id, AmazonAdDebugSettings, CWorks.5cb7c5449e677be888147c58, amobeePasteboard, com.google.maps, com.google.plus.com.deezer.Deezer, com.bmw.a4a.switcher.featureInfos and many more	com.crittercism.uuid, org.OpenUDID,public.utf8-plain-text, com.fiksu.id, public.secureudid, com.google.maps.SSUC, com.flurry.UID, com.bmw.a4a.switcher.featureinfo container,	WiFi MAC Address, 2501110D-69B7-415A-896B-4F7A83591263, ID521411E3-D88E-426E-9B7D-1060C0772C89969DC466, 363046414344413130433230, 8211d087-ca5b-42c3-a1a2-7b3779f6c206, 81C65A17-9F0E-4BFE-83A7-1C2C070C3353, E6644EEB-04B3-4AEF-8562-A2C29E323CCE, 55b0a791-517e-4bd4-8398-414dd527417b, And other binary data instances

Table A.8: List of third-parties knowing names of installed packages on Android (out of a total of 140 Apps tested).

Third-party (Comm type)	Process Names
trademob.com(SSL)	All the processes running on the phone
google.com(SSL)	All the processes running on the phone
google-analytics.com(SSL)	com.anydo, com.rechild.advancedtaskkiller, com.spotify.mobile.android.ui, com.google.android.googlequicksearchbox, com.dailymotion.dailymotion, com.aa.android, com.comuto, com.airbnb.android
doubleclick.net(plain-text)	com.tagdroid.android, com.rechild.advancedtaskkiller, bbc.mobile.news.ww, ua.in.android_wallpapers.spring_nature
crashlytics.com(SSL)	com.evernote, com.path, com.lslk.sleepbot, com.twitter.android, com.dailymotion.dailymotion

Table A.9: List of third-parties knowing names of installed packages on iOS (out of a total of 140 Apps tested).

Third-party (Comm type)	Process Names
flurry.com(plain-text)	TopEleven, Bible, RATP, Transilien, TripIt, DespicableMe, FlyAirIndia, Viadeo, Bankin', VDM, OCB, DuplexA86, SleepBot, Snapchat, Appygraph, Booking.com, foodspotting, Badoo, EDF-Releve, WorldCup2011, Quora, UrbanDictionary, babbelSpanish, MyLittleParis, Volkswagen
google-analytics.com(SSL)	InstantBeautyProduction, Evernote, LILIGO, Transilien, Viadeo, VDM, comuto, easyjet, VintedFR, Volkswagen
crashlytics.com(SSL)	dailymotion, TopEleven, AmazonFR, Path, RunKeeper, foodspotting, babbelSpanish, Deezer
urbanairship.com(SSL)	Wimbledon, RATP, HootSuite, DuplexA86, Appygraph, foodspotting, Volkswagen
xiti.com(plain-text)	laposte, ARTE, myTF1, lequipe, SoundCloud, 20minv3, Leboncoin
admob.com(plain-text)	VSC, BBCNews, WorldCup2011, RF12, UrbanDictionary
capptain.com(plain-text)	Viadeo, myTF1, rtl-fr-radios, 20minv3, iDTGV
tapjoy.com(SSL)	TopEleven, Bible, DespicableMe, OCB, MCT

Appendix B

Measuring Re-identification Risk posed by Smartphone Apps

B.1 Proof of Theorem 1

Metropolis-Hastings algorithm Consider an ergodic Markov chain $\mathcal{G}$ with transition graph $G(\Lambda, E)$ and transition matrix $P_{\mathcal{G}}$, where Λ is the finite state space. For each $x \in \Lambda$, let $\kappa : \Lambda \times \Lambda \to [0,1]$ denote a (not necessarily symmetric) proposal probability distribution function such that for all $x \in \Lambda$, $\kappa(x,x) + \sum_{y \neq x} \kappa(x,y) = 1$. The transitions of $\mathcal{G}$ are defined according to the *Metropolis-Hastings rule* as follows. From any state $x \in \Lambda$, first select an $y \in \Lambda$ such that $(x,y) \in E$ with probability $\kappa(x,y)$. Then, "accept" the transition from x to y with probability $\min\left(1, \frac{\pi(y)\kappa(y,x)}{\pi(x)\kappa(x,y)}\right)$, otherwise stay at x. It is not difficult to show [144] that such a Markov chain $\mathcal{G}$ is reversible, i.e., $\pi(x)P_{\mathcal{G}}(x,y) = \pi(y)P_{\mathcal{G}}(y,x)$ and therefore its stationary distribution π is unique [144]. Consequently, after sufficiently many transitions, the distribution of states will be very close to π. Notice that there is no need to compute the normalization constant of π, even if $|\Omega|$ is very large (i.e., an exponential function of K in our problem), because it appears both in the numerator and denominator of the transition probabilities.

Proof of Theorem 1. In each iteration, $\mathcal{M}$ can select any individual u in D. Hence, at any state, $\mathcal{M}$ can visit any state in Ω^K. Therefore, $\mathcal{M}$ is connected and aperiodic. Also notice that

$$\begin{aligned}
\frac{\pi(C)\kappa(C,S)}{\pi(S)\kappa(S,C)} &= \frac{\kappa(C,S)}{\kappa(S,C)} \\
&= \frac{\sum_{\forall u: U_u \supseteq S} 1/\binom{|U_u|}{K}}{\sum_{\forall u: U_u \supseteq C} 1/\binom{|U_u|}{K}} \\
&= \frac{\sum_{\forall u: U_u \supseteq S} K!/|U| \prod_{i=1}^{K} \frac{1}{|U_u|-K+i}}{\sum_{\forall u: U_u \supseteq C} K!/|U| \prod_{i=1}^{K} \frac{1}{|U_u|-K+i}} \\
&= q(S)/q(C)
\end{aligned}$$

where π is the uniform distribution over Ω^K. Therefore, a candidate next state is accepted with probability $\min\left(1, \frac{\pi(C)\kappa(C,S)}{\pi(S)\kappa(S,C)}\right) = \min\left(1, q(S)/q(C)\right)$, which means that $\mathcal{M}$ is reversible and its unique stationary distribution is π according to the Metropolis-Hastings rule. □

B.2 Proof of Theorem 2

In order to prove $\mathcal{M}$'s mixing time, we use a standard coupling argument which is described below.

Definition 4 (Coupling)**.** *A coupling of a Markov chain $\mathcal{M}$ on state space Ω is a Markov chain on $\Omega \times \Omega$ defining a stochastic process $(X_t, Y_t)_{t=0}^{\infty}$ such that*

- *each of the processes $(X_t, \cdot)$ and $(\cdot, Y_t)$, viewed in isolation, is a faithful copy of the Markov chain $\mathcal{M}$ (given initial states $X_0 = x$ and $Y_0 = y$); that is, $Pr[X_{t+1} = b | X_t = a] = P_{\mathcal{M}}(a, b) = Pr[Y_{t+1} = b | Y_t = a]$; and*
- *if $X_t = Y_t$, then $X_{t+1} = Y_{t+1}$.*

Condition 1 ensures that each process, viewed in isolation, is just simulating the original chain $\mathcal{M}$, and the coupling is designed such that X_t and Y_t tend to coalesce (i.e., move closer to each other according to some notion of distance). Once they meet, Condition 2 guarantees that they will move together forward. The time of this coalescence can be used to upper bound the mixing time which is shown by the next lemma.

Lemma 1 (Coupling lemma [144])**.** *Let $(X_t, Y_t)_{t=0}^{\infty}$ be a coupling of a Markov chain $\mathcal{M}$. For initial states x, y let $T^{x,y} = \min\{t : X_t = Y_t | X_0 = x, Y_0 = y\}$ denote the random variable describing the time until X_t and Y_t coalesce. Then*

$$||P_{\mathcal{M}}^t - \pi||_{tv} \leq \max_{x,y \in \Omega} Pr[T^{x,y} > t]$$

of Theorem 2. Define a coupling (X_t, Y_t) as follows. Let X_t and Y_t choose the same individual u and subset C in Line 6 and 7 of Algorithm 1, respectively. This is a valid coupling according to Definition 4, since both X_t and Y_t are the exact copies of $\mathcal{M}$, and they move together after they coalesce.

Let $p(x) = \kappa(\cdot, x)$ denote the probability that $x = C$ is selected in Line 6 of Algorithm 1. Let $X_0 = x$ and $Y_0 = y$, and, w.l.o.g., $p(x) \leq p(y)$. Due to the coupling rule, X_t and Y_t can coalesce at any time, since $P_{\mathcal{M}}(x, y) > 0$ for all $x, y \in \Omega$. This happens when both X_t and Y_t select a state z such that $p(z) \leq p(x) \leq p(y)$, since $q(z) \leq q(x) \leq q(y)$ will also hold. Let $U_{\max} = \max_u U_u$. For any $x, z \in \Omega$, where z occurs only in $U_{\max}$, $p(z) \leq p(x)$. Indeed,

$$\begin{aligned} p(x) &= \frac{1}{|U|} \sum_{\forall u : U_u \supseteq x} \frac{1}{\binom{|U_u|}{K}} \\ &\geq \frac{1}{|U|} \frac{1}{\binom{|U_{\max}|}{K}} \\ &\geq p(z) \end{aligned}$$

Hence, X_t and Y_t coalesce as soon as they select any $z \in \Omega$ which occur only in the largest record in D. Therefore,

$$\begin{aligned} ||P_{\mathcal{M}}^t - \pi||_{tv} &\leq \max_{x,y \in \Omega^K} Pr[T^{x,y} > t] && \text{(by Lemma 1)} \\ &\leq \sum_{i=t}^{\infty} \left(1 - H_1^*/|U|\right)^i H_1^*/|U| \\ &\leq \left(1 - H_1^*/|U|\right)^t \\ &\leq \exp\left(-t H_1^*/|U|\right) \end{aligned}$$

which proves the theorem. □

Appendix C

MyAdChoices

C.1 Feasibility Problem

This appendix proves the feasibility of the optimization problems (5.2) and (5.4). In particular, it shows that the constraints given by the polyhedron $\mathcal{P}$ are consistent, or said otherwise, that the set of points satisfying them is nonempty. For notational simplicity, we rename the tuples $p^{\min}$ and $p^{\max}$ simply with the symbols r and s, respectively.

For (5.2) and (5.4) to be feasible, we require $\sum_i r_i \leqslant 1$ and $\sum_i s_i \geqslant 1$. To check this, consider the opposite. On the one hand, having $\sum_i r_i > 1$ and $\sum_i s_i < 1$ leads us to a contradiction, since by definition $r_i \preceq s_i$. On the other hand, it is straightforward to verify that, if $\sum_i r_i > 1$, then $\sum_i p_i > 1$, and that, if $\sum_i s_i < 1$, then $\sum_i p_i < 1$, which contradict the fact that p is a PMF.

Next, we prove that the requirement $\sum_i s_i \geqslant 1$ is satisfied. The proof of the condition $\sum_i r_i \leqslant 1$ proceeds along the same lines and is omitted. Recall from Sec. 5.3.2 that the uncertainty class $\mathcal{P}$ is computed by considering an incremental model on the clickstream. That is, each time the user visits a Web page, a new estimate for p is computed from all the pages visited so far. Then, based on this newly estimated distribution, our system updates r and s, if necessary.

The proposed system requires a minimum number of visited pages $w_{\min}$ to estimate p. Following the notation introduced in Sec. 5.4.2, we denote by m_i the number of pages that are classified into the topic category i. When such requirement is met, the tuples r and s are initialized to $r_i = s_i = \frac{m_i}{w_{\min}}$ for all $i = 1, \ldots, n$. In other words, r and s become the MLE of p.

Let s_i^m be the i-th component of the tuple s that results after having visited m pages. It is immediate to check that $s_i^{w_{\min}} \leqslant \cdots \leqslant s_i^m$ is a non-decreasing sequence for all i, which implies that $\sum_i s_i \geqslant 1$. This proves the feasibility of the problems (5.2) and (5.4).

C.2 Linear-Program Formulation of the Robust Minimax Detector

Following the methodology developed by [82, 145], this appendix shows the LP formulation of the robust minimax design problem (5.2). From the definitions of P_i^w and M_{ii}^w, it is easy to verify that (5.2) is equivalent to

$$\max \min_{i=1,2} \inf_{p \in \mathcal{P}} M_{ii},$$

and hence equivalent to the optimization problem

$$\begin{aligned} \text{maximize} \quad & \zeta \\ \text{subject to} \quad & \inf\{\tilde{d}^{\mathrm{T}} p : p \in \mathcal{P}\} \geqslant \zeta, \\ & 1 - \tilde{d}^{\mathrm{T}} q \geqslant \zeta, \\ & 0 \preceq \tilde{d} \preceq \mathbf{1}. \end{aligned} \tag{C.1}$$

Because the primal problem (5.2) is feasible, Slater's constraint qualification is satisfied and therefore strong duality holds for the Lagrange dual problem associated to the linear program (C.1). The dual problem in question is

$$\begin{aligned} \text{maximize} \quad & \mu^{\mathrm{T}} p^{\min} - \lambda^{\mathrm{T}} p^{\max} + \nu \\ \text{subject to} \quad & \mu - \lambda + \nu \mathbf{1} \preceq \tilde{d}, \\ & \lambda \succeq 0, \mu \succeq 0, \end{aligned}$$

where λ, μ, ν are the Lagrange multiplier vectors associated with the minimization problem (C.1), and $p^{\min}, p^{\max}$ determine the polyhedron $\mathcal{P}$ defined in (5.1). Leveraging on this dual problem, we immediately derive the LP formulation (5.3).

Appendix D

MyTrackingChoices

Table D.2 shows the list of frequent trackers overall and in some of the sensitive category web pages. For each third-party" domain, we show the percentage of total web pages (browsed by all users) where it was present. We find that, irrespective of the categories of web pages, Google and Facebook domains as third-party domains are prevalent.

Table D.1: Top 40 ad domains that delivered ads through iframes

doubleclick.net 35.62%	googlesyndication.com 28.81%	weborama.fr 3.80%	2mdn.net 3.66%	rubiconproject.com 2.63%
criteo.com 2.22%	outbrain.com 2.22%	gemius.pl 2.14%	betrad.com 1.78%	pubmatic.com 1.46%
openx.net 1.35%	criteo.net 1.33%	serving-sys.com 0.92%	casalemedia.com 0.79%	bluekai.com 0.70%
amazon-adsystem.com 0.69%	effectivemeasure.net 0.66%	zedo.com 0.54%	adnxs.com 0.53%	tribalfusion.com 0.41%
demdex.net 0.38%	mathtag.com 0.36%	lijit.com 0.35%	adverticum.net 0.34%	infolinks.com 0.34%
doubleverify.com 0.33%	quantserve.com 0.33%	advertising.com 0.32%	mdadx.com 0.31%	flashtalking.com 0.29%
exoclick.com 0.28%	smartadserver.com 0.27%	admission.net 0.20%	ligatus.com 0.20%	sitemeter.com 0.18%
atwola.com 0.17%	teads.tv 0.17%	adition.com 0.16%	rfihub.com 0.13%	atdmt.com 0.11%

Table D.2: Top-10 third-party domains present overall and on some sensitive category web pages

Overall	fbcdn.net 25.55% doubleclick.net 20.44% google-analytics.com 17.77% akamaihd.net 13.78% www.google.com 13.45% gstatic.com 12.62% accounts.google.com 9.74% googleadservices.com 8.97% googlesyndication.com 8.41% apis.google.com 8.38%	**Adult**	google-analytics.com 47.06% youtube.com 23.52% googleadservices.com 21.42% doubleclick.net 19.59% googlesyndication.com 17.16% twitter.com 15.62% apis.google.com 13.29% www.google.com 12.22% googlevideo.com 11.51% ytimg.com 9.82%	**Religion**	gstatic.com 44.80% www.google.com 42.5% apis.google.com 42.30% google-analytics.com 11.53% amazonaws.com 7.69% facebook.com 5.38% youtube.com 4.23% ytimg.com 3.65% xiti.com 3.27% doubleclick.net 3.07%
Health & Fitness	google-analytics.com 24.37% youtube.com 13.89% maps.google.com 13.50% fonts.net 13.19% ytimg.com 13.12% facebook.com 10.63% amazonaws.com 10.09% maps.googleapis.com 8.30% doubleclick.net 6.75% facebook.net 6.05%	**Military**	gstatic.com 19.13% apis.google.com 18.39% plus.google.com 17.58% googleusercontent.com 17.25% googleadservices.com 17.18% www.google.com 16.37% google-analytics.com 8.15% accounts.google.com 4.11% doubleclick.net 3.77% play.google.com 3.36%	**Personal Finance**	google-analytics.com 26.37% doubleclick.net 21.82% cloudfront.net 15.98% googleadservices.com 14.23% scorecardresearch.com 14.00% googlesyndication.com 12.71% googletagservices.com 12.01% facebook.com 11.66% google.co.in 10.50% facebook.net 9.21%

Bibliography

[1] A Primer on Information Theory and Privacy. https://www.eff.org/deeplinks/2010/01/primer-information-theory-and-privacy.

[2] Adblock plus. https://adblockplus.org. accessed on 2016-02-22.

[3] Apple's Latest Crackdown: Apps Pulling The Advertising Identifier, But Not Showing Ads, Are Being Rejected From App Store. http://goo.gl/zWnLuw.

[4] Clickstream or clickpath analysis. accessed on 2015-03-27.

[5] COIN-OR Interior Point OPTimizer. accessed on 2015-09-17.

[6] COIN-OR linear programming solver. accessed on 2015-09-15.

[7] Consumer opt-out. Technical report, Netw. Advertising Initiative. accessed on 2015-03-19.

[8] Disconnect. https://disconnect.me/. accessed on 2016-02-22.

[9] Divide read_phone_state permission in two to provide more secure android for users. accessed on 2016-07-24.

[10] Eclipse public license - version 1.0. accessed on 2015-09-17.

[11] Ghostery. https://www.ghostery.com. accessed on 2016-02-22.

[12] GNU linear programming kit, (GLPK). accessed on 2015-09-17.

[13] Iab quality assurance guidelines (qag) taxonomy. accessed on 2015-09-11.

[14] launchd apple documentation. http://goo.gl/Ysc7Ek.

[15] Lightbeam. accessed on 2015-09-24.

[16] MobileScope. http://www.evidon.com/mobilescope.

[17] Mobilesubstrate framework. http://iphonedevwiki.net/index.php/MobileSubstrate.

[18] Objective-c runtime environment. http://goo.gl/OQAMqh.

[19] The official easylist website. accessed on 2015-10-22.

[20] Permission model & user privacy refinement in future release. accessed on 2016-07-24.

[21] Privacy badger. https://www.eff.org/fr/privacybadger. accessed on 2016-02-22.

[22] Privacy badger. accessed on 2015-09-24.

[23] Privacy of W3C Vibration API. https://blog.lukaszolejnik.com/privacy-of-w3c-vibration-api/.

[24] Privacy tools: Opting out from data brokers. http://juliaangwin.com/privacy-tools-opting-out-from-data-brokers/.

[25] Read phone state and identity should be two separate permissions. accessed on 2016-07-24.

[26] Real-time bidding protocol - cookie matching. accessed on 2015-10-07.

[27] Real-time bidding protocol - processing the request. accessed on 2015-10-07.

[28] Rip: Adblock plus. http://www.engadget.com/2016/02/12/rip-adblock-plus/.

[29] This recycling bin is following you. Quartz.com. 8 août 2013. http://qz.com/112873/this-recycling-bin-is-following-you/.

[30] Trampoline technique. http://en.wikipedia.org/wiki/Trampoline_(computing).

[31] Twitter and Path uploading user's contacts information to their servers. http://www.theregister.co.uk/2012/02/15/twitter_stores_address_books/.

[32] UIPasteboard Class Reference. http://goo.gl/h1o1qX.

[33] WSJ: What They Know - Mobile. http://blogs.wsj.com/wtk-mobile/.

[34] YourOnlineChoices. http://www.youronlinechoices.com/. [Online; accessed 17-February-2016].

[35] Cisco service control online advertising solution guide. Technical report, Cisco Syst., January 2009.

[36] Evercookie - virtually irrevocable persistent cookies, October 2010.

[37] Topline u.s. web data for march 2010. Technical report, March 2010. accessed on 2015-02-19.

[38] Adblock plus user survey results, part 3. Technical report, Eyeo, December 2011. accessed on 2015-07-11.

[39] The state of online advertising. http://www.adobe.com/aboutadobe/pressroom/pdfs/Adobe_State_of_Online_Advertising_Study.pdf, 2012. accessed on 2016-07-25.

[40] Handbook on European data protection law. http://www.echr.coe.int/Documents/Handbook_data_protection_ENG.pdf, 2014. [Online; accessed 17-February-2016].

[41] US programmatic ad spend tops $10 billion this year, to double by 2016. Tech. rep., eMarketer, October 2014.

[42] A Way to Peace in the Adblock War. https://blogs.harvard.edu/vrm/2015/09/21/a-way-to-peace-in-the-adblock-war/, 2015. [Online; accessed 21-February-2016].

[43] Acceptable Ads Manifesto. https://acceptableads.org/, 2015. [Online; accessed 17-February-2016].

[44] DCN Consumer Ad Block Report. https://digitalcontentnext.org/blog/press/digital-content-next-research-indicates-33-of-consumers-likely-to-try-ad-blocking-software-in-next-three-months/, 2015. [Online; accessed 17-February-2016].

[45] Firefox interest dashboard. https://github.com/mozilla/interest-dashboard/blob/master/refData/IAB.js, November 2015. [Online; accessed 21-February-2016].

[46] Getting LEAN with Digital Ad UX. http://www.iab.com/news/lean/, 2015. [Online; accessed 17-February-2016].

[47] Google DoubleClick ad exchange (AdX) buyer program guidelines, April 2015. accessed on 2015-08-05.

[48] Profiling Adblockers. http://www.globalwebindex.net/blog/profiling-adblockers, 2015. [Online; accessed 17-February-2016].

[49] The cost of Ad Blocking, PageFaire and Adobe Ad Blocking Report. http://downloads.pagefair.com/reports/2015_report-the_cost_of_ad_blocking.pdf, 2015. [Online; accessed 17-February-2016].

[50] Tracking preference expression (DNT). Technical report, August 2015.

[51] Why Millennials Block Ads. http://www.dmnews.com/opinions/why-millennials-block-ads/article/475448/, 2015. [Online; accessed 17-February-2016].

[52] Brave. https://brave.com/, 2016. [Online; accessed 17-February-2016].

[53] Efficient & optional filtering of domains in Recursor 4.0.0. http://blog.powerdns.com/2016/01/19/efficient-optional-filtering-of-domains-in-recursor-4-0-0/, 2016. [Online; accessed 21-February-2016].

[54] List of allowed third-party domains. https://myrealonlinechoices.inrialpes.fr/allowed_thirdparty_domains.json, 2016. [Online; accessed 17-February-2016].

[55] MyTrackingChoices: Available to download at Chrome Web Store. https://chrome.google.com/webstore/detail/mytrackingchoices/fmonkjimgifgcgeocdhhgbfoncmjclka?hl=fr, 2016. [Online; accessed 21-February-2016].

[56] Why do people block online ads? https://www.adspeed.com/Blog/people-block-online-ads-1844.html, 2016. [Online; accessed 17-February-2016].

[57] Why privacy is important, and having "nothing to hide" is irrelevant. https://robindoherty.com/2016/01/06/nothing-to-hide.html, 2016. [Online; accessed 16-July-2016].

[58] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proc. ACM Conf. Comput., Commun. Secur. (CCS)*, pages 674–689, Washington, DC, November 2014.

[59] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 674–689, New York, NY, USA, 2014. ACM.

[60] Jagdish Prasad Achara, Gergely Acs, and Claude Castelluccia. On the Unicity of Smartphone Applications. In *The 14th ACM Workshop on Privacy in the Electronic Society (ACM WPES)*, 2015.

[61] Jagdish Prasad Achara, Mathieu Cunche, Vincent Roca, and Aurélien Francillon. Short Paper: WifiLeaks: Underestimated Privacy Implications of the Access_Wifi_State Android Permission. In *The 7th ACM Conference on Security and Privacy in Wireless & Mobile Networks (ACM WiSec)*, 2014.

[62] Jagdish Prasad Achara, James-Douglass Lefruit, Vincent Roca, and Claude Castelluccia. Detecting privacy leaks in the RATP App: how we proceeded and what we found. *Journal of Computer Virology and Hacking Techniques (Springer JCVHT)*, 2014.

[63] Jagdish Prasad Achara, Javier Parra-Arnau, and Claude Castelluccia. MyTrackingChoices: Pacifying the Ad-Block War by Enforcing User Privacy Preferences. *The 15th Annual Workshop on the Economics of Information Security (WEIS)*, 2016.

[64] Jagdish Prasad Achara, Vincent Roca, Claude Castelluccia, and Aurélien Francillon. MobileAppScrutinator: A Simple yet Efficient Dynamic Analysis Approach for Detecting Privacy Leaks across Mobile OSs. *Submitted to The 32nd Annual Computer Security Applications Conference (ACSAC)*, 2016.

[65] Gergely Acs, Jagdish Prasad Achara, and Claude Castelluccia. Probabilistic Km-anonymity: Efficient Anonymization of Large Set-valued Datasets. In *IEEE International Conference on Big Data (IEEE Big Data)*, 2015.

[66] Yuvraj Agarwal and other. Protectmyprivacy: detecting and mitigating privacy leaks on ios devices using crowdsourcing. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, 2013.

[67] K. G. Allen. Search marketing tops online retail customer acquisition tactics. Technical report, NFR, July 2014. accessed on 2016-01-11.

[68] M. Aly, A. Hatch, V. Josifovski, and V. K. Narayanan. Web-scale user modeling for targeting. In *Proc. Int. WWW Conf.*, pages 3–12. ACM, 2012.

[69] AOL search data scandal, August 2006. accessed on 2013-11-15.

[70] M. Arment. The ethics of modern web ad-blocking, August 2015. accessed on 2015-08-15.

[71] Steven Arzt, Siegfried Rasthofer, E Bodden, A Bartel, J Klein, Y Le Traon, D Octeau, and P McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. In *Proceedings of the 35th annual ACM SIGPLAN conference on Programming Language Design and Implementation (PLDI 2014)*, 2014.

[72] Audience Buying Guide 2011. accessed on 2015-03-27.

[73] P. Barford, I. Canadi, D. Krushevskaja, Q. Ma, and S. Muthukrishnan. Adscape: Harvesting and analyzing online display ads. In *Proc. ACM Int. WWW Conf.*, pages 597–608. ACM, 2014.

[74] S. J. Benson, Y. Ye, and X. Zhang. Solving large-scale sparse semidefinite programs for combinatorial optimization. *(SIAM) J. Optim.*, 10(2):443–461, 2000.

[75] L. Bentivogli, P. Forner, B. Magnini, and E. Pianta. Revising wordnet domains hierarchy: Semantics, coverage, and balancing. In *Proc. PostCOLING Workshop Multiling. Ling. Resources*, pages 101–108, Hangzhou, China, August 2004.

[76] M. Berkelaar, K. Eikland, and P. Notebaert. Open source (mixed integer) linear programming system, May 2004.

[77] Anna Berlee. Using nyc taxi data to identify muslim taxi drivers. `http://www.theiii.org/index.php/997/using-nyc-taxi-data-to-identify-muslim-taxi-drivers/`, 2015. accessed on 2016-07-25.

[78] Jeffrey R Blum, Daniel G Greencorn, and Jeremy R Cooperstock. Smartphone sensor reliability for augmented reality applications. In *Mobile and Ubiquitous Systems: Computing, Networking, and Services*. Springer, 2013.

[79] Theodore Book, Adam Pridgen, and Dan S Wallach. Longitudinal analysis of android ad library permissions. *arXiv preprint arXiv:1303.0857*, 2013.

[80] Theodore Book and Dan S. Wallach. A case of collusion: A study of the interface between ad libraries and their apps. *CoRR*, abs/1307.6082, 2013.

[81] B. Borchers. CSDP, a C library for semidefinite programming. *Optim. Method, Softw.*, 11(1):613–623, 1999.

[82] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[83] J. Butler. Case study: How display ad remarketing works in travel. Technical report, Tnooz, September 2010. accessed on 2016-01-14.

[84] Juan Pablo Carrascal, Christopher Riederer, Vijay Erramilli, Mauro Cherubini, and Rodrigo de Oliveira. Your browsing behavior for a big mac: Economics of personal information online. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 189–200, New York, NY, USA, 2013. ACM.

[85] J. M. Carrascosa, J. Mikians, R. Cuevas, V. Erramilli, and N. Laoutaris. Understanding interest-based behavioural targeted advertising. In *arXiv: 1411.5281v1*, November 2014.

[86] J. M. Carrascosa, J. Mikians, R. Cuevas, V. Erramilli, and N. Laoutaris. I always feel like somebody's watching me. measuring online behavioural advertising. In *Proc. ACM Int. Emerg. Netw. Experiments, Technol. (CoNEXT)*, Heidelberg, Germany, December 2015.

[87] Farah Chanchary and Sonia Chiasson. User perceptions of sharing, advertising, and tracking. In *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*, pages 53–67, 2015.

[88] Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm, 1995.

[89] Greg Conti. *Googling Security: How Much Does Google Know About You?* Addison-Wesley Professional, 1 edition, 2008.

[90] R. Cookson. Google, Microsoft and Amazon pay to get around ad blocking tool. *Financial Times*, February 2015. accessed on 2016-02-22.

[91] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, New York, second edition, 2006.

[92] Jim Cowie and Wendy Lehnert. Information extraction. *Communications of the ACM*, 39(1), 1996.

[93] Mathieu Cunche, Mohamed-Ali Kaafar, and Roksana Boreli. Linking wireless devices using information contained in wi-fi probe requests. *Pervasive and Mobile Computing*, 2013.

[94] J. Currie and D. I. Wilson. OPTI: Lowering the barrier between open source optimizers and the industrial Matlab user. In *Proc. Found. Comput.-Aided Process Oper.*, January 2012.

[95] Cuthbert Daniel and Wilkinson Glenn. Snoopy: Distributed tracking and profiling framework. In *44Con 2012*, 2012.

[96] A. Datta, M. C. Tschantz, and A. Datta. Automated experiments on ad privacy settings. In *Proc. Int. Symp. Priv. Enhanc. Technol. (PETS)*, Darmstadt, Germany, July 2015.

[97] J. Daudé, , L. Padró, and German Rigau. Validation and tuning of wordnet mapping techniques. *Proc. Int. Conf. Recent Adv. Nat. Lang. Process. (RANLP)*, September 2003.

[98] W. Davis. Ftc's julie brill tells ad tech companies to improve privacy protections, September 2015. accessed on 2015-10-01.

[99] Yves-Alexandre de Montjoye, Cesar A. Hidalgo, Michel Verleysen, and Vincent D. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific Reports, Nature*, March 2013.

[100] Yves-Alexandre de Montjoye, Laura Radaelli, Vivek Kumar Singh, and Alex Pentland. Unique in the shopping mall: On the reidentifiability of credit card metadata. *Science*, 347(6221), January 2015.

[101] Peter Eckersley. How unique is your web browser? In *PETS*, 2010.

[102] Manuel Egele, Christopher Kruegel, Engin Kirda, and Giovanni Vigna. PiOS : Detecting privacy leaks in iOS applications. In *NDSS 2011, 18th Annual Network and Distributed System Security Symposium, San Diego, CA, USA*, February 2011.

[103] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol Sheth. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. In *Proc. of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Vancouver, Canada, October 2010.

[104] S. Englehardt. The hidden perils of cookie syncing. August 2014. accessed on 2014-12-10.

[105] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. INRIA Research Report Nř8539, `http://randomwalker.info/publications/OpenWPM_1_million_site_tracking_measurement.pdf`, 2016.

[106] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. [Technical Report], May 2016.

[107] Steven Englehardt, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan Mayer, Arvind Narayanan, and Edward W. Felten. Cookies that give you away: The surveillance implications of web tracking. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 289–299, New York, NY, USA, 2015. ACM.

[108] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. Android permissions demystified. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, New York, NY, USA, 2011. ACM.

[109] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. Android permissions: User attention, comprehension, and behavior. SOUPS '12, New York, NY, USA, 2012. ACM.

[110] E. Ferrari and B. Thuraisingham. chapter Secure Database Systems, pages 353–403. Artech House, Inc., 2000.

[111] J. Q. Freed. Hoteliers rake in returns through retargeting. Technical report, Hotel News Now, March 2012. accessed on 2016-01-14.

[112] Nathaniel Fruchter, Hsin Miao, Scott Stevenson, and Rebecca Balebako. Variations in tracking in relation to geographic location. *CoRR*, abs/1506.04103, 2015.

[113] Adam P Fuchs, Avik Chaudhuri, and Jeffrey S Foster. Scandroid: Automated security certification of android applications. *Manuscript, Univ. of Maryland*, 2009.

[114] S. Gauch, M. Speretta, A. Chandramouli, and A. Micarelli. *The adaptive Web*, chapter User profiles for personalized information access, pages 54–89. Springer-Verlag, 2007.

[115] E. M. Gertz and S. J. Wright. Object-oriented software for quadratic programming. *ACM Trans. Math. Softw.*, 29:58–81, 2003.

[116] John Geweke. Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. In *In Bayesian Statistics*, pages 169–193. University Press, 1992.

[117] Clint Gibler, Jonathan Crussell, Jeremy Erickson, and Hao Chen. Androidleaks: Automatically detecting potential privacy leaks in android applications on a large scale. In *Trust and Trustworthy Computing*. Springer, 2012.

[118] Sharad Goel, Jake M Hofman, and M Irmak Sirer. Who does what on the web: A large-scale study of browsing behavior. In *ICWSM*, 2012.

[119] A. Gonzalez-Agirre, E. Laparra, and G. Rigau. Multilingual central repository version 3.0: upgrading a very large lexical knowledge base. In *Proc. Global WordNet Conf.*, 2012.

[120] 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 *USENIX HotOS workshop*, 2007.

[121] S. Guha, B. Cheng, and P. Francis. Challenges in measuring online advertising systems. In *Proc. ACM Internet Meas. Conf. (IMC)*, Melbourne, Australia, November 2010.

[122] M. Gundlach. AdBlock. `https://getadblock.com/`, 2016. accessed on 2016-02-22.

[123] Seungyeop Han et al. A study of third-party tracking by mobile apps in the wild. Technical report, 2011.

[124] Brian Hayes. Uniquely me! how much information does it take to single out one person among billions? 2014.

[125] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[126] B. J. Jansen. Click fraud. *IEEE Comput.*, 40(7):85–86, July 2007.

[127] E. T. Jaynes. Information theory and statistical mechanics II. *Phys. Review Ser. II*, 108(2):171–190, 1957.

[128] E. T. Jaynes. On the rationale of maximum-entropy methods. *Proc. IEEE*, 70(9):939–952, September 1982.

[129] Jinseong Jeon, Kristopher K Micinski, and Jeffrey S Foster. Symdroid: Symbolic execution for dalvik bytecode. 2012.

[130] S. G. Johnson. NLopt nonlinear-optimization package. accessed on 2015-09-16.

[131] A. Kae, K. Kan, V. K. Narayanan, and D. Yankov. Categorization of display ads using image and landing page features. In *Proc. ICDM Workshop Large-Scale Data Min.: Theory, Appl.*, pages 1–8. ACM, 2011.

[132] Terence Kawaja. Display LUMAscape. `http://www.lumapartners.com/lumascapes/display-ad-tech-lumascape/`. accessed on 2016-07-22.

[133] Jinyung Kim, Yongho Yoon, Kwangkeun Yi, Junbum Shin, and SWRD Center. Scandal: Static analyzer for detecting privacy leaks in android applications. *MoST*, 2012.

[134] P. Kouvelis and G. Yu. *Robust Discrete Optimization and Its Applications*. Springer-Verlag, first edition, November 1996.

[135] Michael Kranch and Joseph Bonneau. Upgrading HTTPS in Mid-Air: An Empirical Study of Strict Transport Security and Key Pinning. In *NDSS '15: The 2015 Network and Distributed System Security Symposium*, February 2015.

[136] Joanna Brenner Kristen Purcell and Lee Rainie. Survey on search engine use. `http://www.pewinternet.org/2012/03/09/main-findings-11/`. accessed on 2016-02-22.

[137] Ravi Kumar and Andrew Tomkins. A characterization of online browsing behavior. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 561–570, New York, NY, USA, 2010. ACM.

[138] Anthony LaMarca, Yatin Chawathe, Sunny Consolvo, Jeffrey Hightower, Ian Smith, James Scott, Timothy Sohn, James Howard, Jeff Hughes, Fred Potter, et al. Place lab: Device positioning using radio beacons in the wild. In *Pervasive computing*. Springer, 2005.

[139] Yuan Tian Le Nguyen, Sungho Cho, Wookjong Kwak, Sanjay Parab, Yuseung Kim, Patrick Tague, and Joy Zhang. Unlocin: Unauthorized location inference on smartphones without being caught.

[140] M. Lecuyer, G. Ducoffe, F. Lan, A. Papancea, T. Petsios, R. Spahn, A. Chaintreau, and R. Geambasu. XRay: Enhancing the web's transparency with differential correlation. In *Proc. Conf. USENIX Secur. Symp.*, August 2014.

[141] M. Lecuyer, R. Spahn, Y. Spiliopoulos, A. Chaintreau, R. Geambasu, and D. Hsu. Sunlight: finegrained targeting detection at scale with statistical confidence. In *Proc. ACM Conf. Comput., Commun. Secur. (CCS)*, August 2015.

[142] Daniel Lemire, Leonid Boytsov, and Nathan Kurz. SIMD compression and the intersection of sorted integers. *CoRR*, abs/1401.6399, 2014.

[143] Pedro Giovanni Leon, Blase Ur, Yang Wang, Manya Sleeper, Rebecca Balebako, Richard Shay, Lujo Bauer, Mihai Christodorescu, and Lorrie Faith Cranor. What matters to users?: Factors that affect users' willingness to share information with online advertisers. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*, SOUPS '13, pages 7:1–7:12, New York, NY, USA, 2013. ACM.

[144] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 2009.

[145] B. C. Levy. *Principles of Signal Detection and Parameter Estimation*. Springer-Verlag, first edition, 2008.

[146] Janne Lindqvist, Tuomas Aura, George Danezis, Teemu Koponen, Annu Myllyniemi, Jussi Mäki, and Michael Roe. Privacy-preserving 802.11 access-point discovery. In *ACM WiSec*, 2009.

[147] B. Liu, A. Sheth, U. Weinsberg, J. Chandrashekar, and R. Govindan. Adreveal: Improving transparency into online targeted advertising. In *Proc. Hot Topics in Netw.*, pages 12:1–12:7. ACM, 2013.

[148] R. Lougee-Heimer. The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community. *IMB J. Res. Develop.*, 47(1):57–66, January 2003.

[149] Long Lu, Zhichun Li, Zhenyu Wu, Wenke Lee, and Guofei Jiang. Chex: statically vetting android apps for component hijacking vulnerabilities. In *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012.

[150] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. *L*-diversity: Privacy beyond *k*-anonymity. *TKDD*, 1(1), 2007.

[151] B. Magnini and G. Cavaglià. Integrating subject field codes into wordnet. In *Proc. Lang. Resource, Evaluation (LREC)*, pages 1413–1418, June 2000.

[152] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999.

[153] Don Marti. Targeted Advertising Considered Harmful. `http://zgp.org/targeted-advertising-considered-harmful/`. accessed on 2016-02-22.

[154] G. Marvin. Consumers now notice retargeted ads. Technical report, Marketing Land, December 2013. accessed on 2015-08-12.

[155] Célestin Matte, Jagdish Prasad Achara, and Mathieu Cunche. Device-to-identity Linking Attack Using Targeted Wi-fi Geolocation Spoofing. In *The 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks (ACM WiSec)*, 2015.

[156] Jonathan R. Mayer and other. Third-party web tracking: Policy and technology. SP '12, 2012.

[157] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, (6):1087–1092.

[158] G. A. Miller. WordNet: A lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.

[159] T. Morey, T. Forbath, and A. Schoop. Customer data: Designing for transparency and trust. Internet draft, May 2015. accessed on 2014-02-18.

[160] K. Mowery and H. Shacham. Pixel perfect: Fingerprinting canvas in HTML5. In *Proc. IEEE Web 2.0 Workshop Secur., Priv. (W2SP)*. IEEE Comput. Soc., May 2012.

[161] A. B. M. Musa and Jakob Eriksson. Tracking unmodified smartphones using Wi-Fi monitors. In *ACM SenSys'12*, 2012.

[162] J. Naughton. The rise of ad-blocking could herald the end of the free internet, September 2015. accessed on 2015-10-04.

[163] T.-D. Nguyen. *Robust Estimation, Regression and Ranking with Applications in Portfolio Optimization*. PhD thesis, MIT, June 2009.

[164] L. Olejnik. *Measuring the Privacy Risks and Value of Web Tracking*. PhD thesis, Nat. Inst. Res. Comput. Sci., Contr. (INRIA), January 2015.

[165] L. Olejnik, T. Minh-Dung, and C. Castelluccia. Selling off privacy at auction. In *Proc. IEEE Symp. Netw. Distrib. Syst. Secur. (SNDSS)*, February 2014.

[166] Lukasz Olejnik. Report on sensors apis: privacy and transparency perspective. 2016.

[167] Lukasz Olejnik, Gunes Acar, Claude Castelluccia, and Claudia Diaz. The leaking battery: A privacy analysis of the html5 battery status api. 2016.

[168] Lukasz Olejnik, Claude Castelluccia, and Artur Janc. On the uniqueness of web browsing history patterns. *Annals of Telecommunications*, 69(1), February 2014.

[169] Lukasz Olejnik, Tran Minh-Dung, and Claude Castelluccia. Selling off privacy at auction. 2013.

[170] Adam J. Oliner, Anand P. Iyer, Ion Stoica, Eemil Lagerspetz, and Sasu Tarkoma. Carat: Collaborative energy diagnosis for mobile devices. In *ACM SenSys*, 2013.

[171] PageFaire. The cost of ad blocking. `https://downloads.pagefair.com/wp-content/uploads/2016/05/2015_report-the_cost_of_ad_blocking.pdf`, 2015. accessed on 2016-07-25.

[172] S. Pandey, M. Aly, A. Bagherjeiran, A. Hatch, P. Ciccolo, A. Ratnaparkhi, and M. Zinkevich. Learning to target: What works for behavioral targeting. In *Proc. Int. Conf. Inform., Knowl. Manage. (CIKM)*, pages 1805–1814. ACM, 2011.

[173] J. Parra-Arnau, D. Rebollo-Monedero, and J. Forné. Measuring the privacy of user profiles in personalized information systems. *Future Gen. Comput. Syst. (FGCS), Special Issue Data, Knowl. Eng.*, 33:53–63, April 2014.

[174] Javier Parra-Arnau, Jagdish Prasad Achara, and Claude Castelluccia. MyAdChoices: Bringing Transparency and Control to Online Advertising. *Submitted to The ACM Transactions on the Web (ACM TWEB)*, 2016.

[175] Javier Parra-Arnau, Jagdish Prasad Achara, and Claude Castelluccia. MyAdChoices: Bringing Transparency and Control to Online Advertising. *arXiv preprint arXiv:1602.02046*, 2016.

[176] S. Puglisi, D. Rebollo-Monedero, and J. Forné. You never surf alone. ubiquitous tracking of users browsing habits. In *Proc. Int. Workshop Data Priv. Manage. (DPM)*, Lecture Notes Comput. Sci. (LNCS), Vienna, Austria, September 2015.

[177] K. Purcell, J. Brenner, and Lee Rainie. Search engine use 2012. Res. rep., Pew Internet, Amer. Life Project, March 2012.

[178] Abbas Razaghpanah, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Christian Kreibich, Phillipa Gill, Mark Allman, and Vern Paxson. Haystack: In situ mobile traffic analysis in user space. *CoRR*, 2015.

[179] D. Rebollo-Monedero, J. Parra-Arnau, and J. Forné. An information-theoretic privacy criterion for query forgery in information retrieval. In *Proc. Int. Conf. Secur. Technol.(SecTech)*, volume 259 of *Commun. Comput., Inform. Sci. (CCIS)*, pages 146–154, Jeju Island, South Korea, December 2011. Springer-Verlag.

[180] Nicky Robinson and Joseph Bonneau. Cognitive Disconnect: Understanding Facebook Connect Login Permissions. In *COSN '14: ACM Conference on Online Social Networks*, October 2014.

[181] Franziska Roesner and other. Detecting and defending against third-party tracking on the web. NSDI'12.

[182] David Rogers. How business can gain consumers' trust around data. `http://www.forbes.com/sites/davidrogers/2015/11/02/how-business-can-gain-consumers-trust-around-data/#493bcac14de1`, 2015. accessed on 2016-07-22.

[183] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[184] Golam Sarwar and other. On the effectiveness of dynamic taint analysis for protecting against private information leaks on android-based devices. In *10th International Conference on Security and Cryptography (SECRYPT)*, 2013.

[185] P. Sayer. Adblock extension begins whitelisting "acceptable ads", October 2015.

[186] M. J. Schervish. *Theory of Statistics*. Springer-Verlag, New York, 1995.

[187] Naked Security. Path and hipster iphone apps leak sensitive data without notification. `https://nakedsecurity.sophos.com/2012/02/08/apple-mobile-apps-path-and-hipster-and-leak-sensitive-data-without-notification/`, 2012. accessed on 2016-07-25.

[188] Suranga Seneviratne, Aruna Seneviratne, Prasant Mohapatra, and Anirban Mahanti. Predicting user traits from a snapshot of apps installed on a smartphone. *SIGMOBILE Mob. Comput. Commun. Rev.*, June 2014.

[189] M. Smith. *Targeted: How Technology Is Revolutionizing Advertising and the Way Companies Reach Consumers*. AMACOM, New York, first edition, November 2014.

[190] A. Soltani, S. Canty, Q. Mayo, L. Thomas, , and C. J. Hoofnagle. Flash cookies and privacy. In *Proc. AAAI Spring Symp. Intell. Inform. Priv. Manage.* Assoc. Adv. Artif. Intell., 2010.

[191] Oleksii Starov, Johannes Dahse, Syed Sharique Ahmad, Thorsten Holz, and Nick Nikiforakis. No honor among thieves: A large-scale analysis of malicious web shells. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 1021–1032, Republic and Canton of Geneva, Switzerland, 2016. International World Wide Web Conferences Steering Committee.

[192] Latanya Sweeney. Simple demographics often identify people uniquely. 2000.

[193] Latanya Arvette Sweeney. *Computational Disclosure Control: A Primer on Data Privacy Protection.* PhD thesis, Cambridge, MA, USA, 2001.

[194] S. Thielman. Rise of ad-blockers shows advertising does not understand mobile, say experts, October 2015. accessed on 2015-10-05.

[195] NYC TLC. Tlc trip record data. `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`, 2015. accessed on 2016-07-25.

[196] V. Toubiana. SquiggleSR, 2007.

[197] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas. Adnostic: Privacy preserving targeted advertising. In *Proc. IEEE Symp. Netw. Distrib. Syst. Secur. (SNDSS)*, pages 1–21, February 2010.

[198] Hien Thi Thu Truong, Eemil Lagerspetz, Petteri Nurmi, Adam J. Oliner, Sasu Tarkoma, N. Asokan, and Sourav Bhattacharya. The company you keep: Mobile malware infection rates and inexpensive risk indicators. In *WWW*, 2014.

[199] M. M. Tsang, S. C. Ho, and T. P. Liang. Consumer attitudes toward mobile advertising: An empirical study. *Int. J. Electron. Commer.*, 8(3):65–78, 2004.

[200] Joseph Turow, Jennifer King, Chris Jay Hoofnagle, Amy Bleakley, and Michael Hennessy. Americans reject tailored advertising and three activities that enable it (september 29, 2009). `http://ssrn.com/abstract=1478214` or `http://dx.doi.org/10.2139/ssrn.1478214`. accessed on 2016-02-22.

[201] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Math. Program.*, 106(1):25–57, 2006.

[202] V. Woollaston. Facebook slammed after advertising funeral directors to a cancer patient, March 2015. accessed on 2015-05-20.

[203] J. Yan, N. Liu, G. Wang, W. Zhang, Y. Jiang, and Z. Chen. How much can behavioral targeting help online advertising? In *Proc. Int. WWW Conf.*, pages 261–270. ACM, 2009.

[204] K. Yang, Y. Wu, J. Huang, X. Wang, and S. Verdu. Distributed robust optimization for communication networks. In *Proc. Joint Conf. IEEE Comput., Commun. Soc. (INFOCOM)*, 2008.

[205] Zhemin Yang, Min Yang, Yuan Zhang, Guofei Gu, Peng Ning, and X Sean Wang. Appintent: Analyzing sensitive data transmission in android for privacy leakage detection. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security.*

[206] S. Yuan, A. Z. Abidin, M. Sloan, and J. Wang. Internet advertising: An interplay among advertisers, online publishers, ad exchanges and web users. *arXiv: 1206.1754*, 2012. arXiv preprint.

[207] ZDNet. Twitter uploads contact list data without consent; retains for 18 months. `http://www.zdnet.com/article/twitter-uploads-contact-list-data-without-consent-retains-for-18-months/`, 2012. accessed on 2016-07-25.

[208] Xiaoyong Zhou, Soteris Demetriou, Dongjing He, Muhammad Naveed, Xiaorui Pan, XiaoFeng Wang, Carl A. Gunter, and Klara Nahrstedt. Identity, location, disease and more: Inferring your secrets from android public resources. In *ACM CCS 2013*.

[209] C. Zhu, R. H. Byrd, and J. Nocedal. L-bfgs-b: Algorithm 778: L-bfgs-b fortran routines for large scale bound constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, 2007.

[210] A. M. Zoubir, V. Koivunen, and Y. Chakhchoukh M. Muma. Robust estimation in signal processing: A tutorial-style treatment of fundamental concepts. *IEEE Signal Process. Mag.*, 29(4):61–80, July 2012.