Intrinsically Inseparable: Investigating Novel Tracking Practices and Assessing the Carbon Footprint of Ads

Naif Mehanna

Université de Lille

Inria
INVENTEURS DU MONDE NUMÉRIQUE

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille

Doctor of Philosophy in Computer Science

MADIS Doctoral School
Specialization : Computer Science

Naif Mehanna

University of Lille & CRIStAL & Inria
Spirals team

Intrinsically Inseparable: Investigating Novel Tracking Practices and Assessing the Carbon Footprint of Ads

Thesis defended on May 30, 2024, in front of a jury composed of

Rapporteurs:	Kévin Huguenin	-	Professor at the University of Lausanne
	Sascha Fahl	-	Professor at the University of Hannover
Éxaminateurs:	Oana Goga	-	Researcher at the CNRS
	Benjamin Nguyen	-	Professor at INSA Centre Val de Loire
Directeur de thèse:	Walter Rudametkin	-	Professor at the University of Rennes
President:	Jeremie Dequidt	-	Professor at the University of Lille

Doctorat en Informatique

École Doctorale MADIS
Spécialité : Informatique

Naif Mehanna

Université de Lille & CRIStAL & Inria
Équipe-Projet Spirals

Intrinsèquement Inséparables : Enquête sur les Nouvelles Pratiques de Suivi et Évaluation de l'Empreinte Carbone des Publicités

Thèse soutenue le 30 Mai 2024 devant le jury composé de

Rapporteurs:	Kévin Huguenin	-	Professeur à l'Université de Lausanne
	Sascha Fahl	-	Professeur à l'Université de Hanovre
Examinateurs:	Oana Goga	-	Chargée de Recherche au CNRS
	Benjamin Nguyen	-	Professeur à l'INSA Centre Val de Loire
Directeur de thèse:	Walter Rudametkin	-	Professeur à l'Université de Rennes
Président:	Jeremie Dequidt	-	Professeur à l'Université de Lille

Acknowledgements

I started this thesis with many doubts and questions in September 2020. Four years later, I am now realizing how this experience has thoroughly changed the way I function thanks to the multiple amazing encounters and the work I pursued. However, this thesis would never have been possible without the invaluable help and presence of many people.

The first person I would like to thank and express my gratitude to is my supervisor, Walter Rudametkin, without whom I would never have pursued this thesis. Thanks for believing in me, even in times when I didn't believe in myself. Thank you for being more than a supervisor during all the time that we've known each other and becoming a friend I could count on.

Obviously, my academic journey would not have been the same without all the people in the Spirals team. Thanks to the professors in the team for always being of good counsel and helping me the best they could when I needed it. Thanks to Lionel Seinturier for being a great team leader; your work ethic is an example that everybody should look up to. I would like to give special thanks to Pierre Laperdrix and Romain Rouvoy, your work has played a large part in what motivated me to join the Spirals team. Working with both of you has been great, and I learned a lot from you. Thanks to Guillaume Fieni for all the late-night help and for getting me out of hot water before paper deadlines. Thanks to Maxime, Pierre J., Sihem, Imane, Maryam, and all the others in the team for the great years together, the help and support, and all the interesting talks we've had. Thanks to Salman Farhat for the laughs, the support, and for being a great friend all along.

Finally, and most importantly, I would like to emphasize how nothing would have been possible without the constant support from my family. You are the reason that keeps my motivation up and high every day.
Thanks to my mother for always pushing me to be the best person I can be, whether through my work or in everyday life. I would never have succeeded at this PhD if it were not for your motivation, your constant guidance, and your presence. Thanks to my father for believing in my achievements and seeing the best in me, I would not have been where I stand now if it weren't for you too.
Thanks to my grandparents for always being present and constantly celebrating my successes as if they were their own.
Thanks to my life partner, Louisa, who had to endure my mood swings and the inevitable failures, the times of self-doubt, but despite it all, always managed to find the right words and be present in the joyful moments.
Finally, I would like to thank my two younger brothers: you both played the role of a big brother when I needed you, and I could not have achieved anything without the laughs and talks we had and continue to have.

Naif Mehanna, Lille, July 2024.

Abstract

Online advertising has become a major component of today's Web, fueled by advanced tracking mechanisms and profiling algorithms. The boom of social media in the mid-2000s and the exponential use of smartphones have contributed to turning online advertising into an industry worth over $US600 Billion. However, growing scrutiny on online privacy has led major entities to introduce defenses against online tracking: this is showcased by the introduction of regional regulations, such as the GDPR, and by defenses included in Web browsers, that limit the tracking capabilities of advertisers. This increased scrutiny has led advertisers to investigate novel tracking mechanisms. Recent years saw the growth of browser fingerprinting as a tracking technique, which exploits the software and hardware's diversity of devices roaming the Web. This constant rush for more precise tracking is not without consequences and amplifies the online advertising industry's demand for data centers, processing power, and better network infrastructure, to collect a constantly growing flow of data and distribute increasingly richer content, increasing the environmental impact of online advertising.

In this thesis, I shed light on different aspects of the online tracking and advertising ecosystem by introducing three contributions.

1. First, we introduce DrawnApart, a technique that leverages unique properties in the GPU stack to identify individual devices. Using DrawnApart, we show that we are capable of differentiating between identical devices, with success rates reaching over 95% in a controlled environment. Through a crowd-sourced data collection on the AmIUnique platform, we evaluated our technique on $2,550$ unique devices and $370,392$ fingerprints and found that it can significantly extend the tracking time of the state-of-the-art fingerprint-based tracking algorithm, FP-Stalker.

2. As smartphones have found their way into the pockets of the vast majority of the world's population, we investigated the tracking ecosystem of games on Android. Our analysis of a dataset comprised of $6,355$ free games and 396 paid games shows that paid games are effectively less prone to online tracking compared to their free counterpart, but were still subject to a significant amount of tracking in some cases. We also investigated the *Teachers Approved* program and found that it significantly reduced the number of trackers and advertising in games, highlighting the importance of enforcement on the privacy ecosystem.

3. Finally, we introduce an end-to-end measurement methodology of the carbon impact of the online advertising industry. We find that browsing the Web while allowing all types of trackers and advertising leads to a 144% increase in carbon emissions. Our results also show that the impact of guidelines and regulations can have positive environmental effects: we instrumented cookie banners and found that refusing cookies can lead to a significant decrease in the carbon impact of the browsing session.

Resumé

Alimentée par des méthodologies de traçage complexes et avancées, la publicité en ligne est devenue un élément majeur du Web tel que nous le connaissons aujourd'hui. L'explosion de l'usage des réseaux sociaux au milieu des années 2000 et l'adoption exponentielle des smartphones ont contribué à transformer le domaine de la publicité en ligne en une industrie valant plus de 600 milliards de dollars US. Cependant, les problématiques de vie privée sur le Web prenant de l'ampleur depuis quelques années ont poussé les acteurs principaux du Web à introduire des défenses contre le traçage en ligne. Des régulations régionales, comme le RGPD, et des défenses directement incluses dans les navigateurs poussent ainsi les publicitaires à rechercher de nouvelles méthodologies de suivi, moins connues mais maintenant une efficacité élevée. C'est notamment le cas du traçage par empreinte de navigateur, qui exploite la configuration logicielle et matérielle des appareils naviguant le Web pour les identifier. Cette course perpétuelle vers un suivi toujours plus précis n'est pas sans conséquences et amplifie la demande de l'industrie de la publicité en ligne pour une infrastructure Web toujours plus capable et toujours plus gourmande en ressources afin d'accommoder un flux de données en augmentation constante et des publicités toujours plus riches en contenu, ce qui contribue à augmenter l'impact environnemental de la publicité en ligne.

À travers cette thèse, je mets en lumière différents aspects du traçage en ligne et de l'industrie de la publicité en introduisant trois contributions majeures :

1. Dans un premier temps, nous introduisons DrawnApart, une technique qui exploite des propriétés uniques des cartes graphiques afin d'identifier des appareils identiques. Nous montrons ainsi que notre algorithme est capable de différencier des appareils avec la même configuration logicielle et matérielle, avec des taux de succès dépassant les 95% en environnement contrôlé. Nous évaluons également DrawnApart sur 2 550 appareils uniques et 370 392 empreintes de navigateurs collectés sur la plateforme AmIUnique et montrons que nous sommes capables d'allonger de manière significative le temps de traçage par empreinte de navigateur de l'état de l'art, FP-Stalker.

2. Les smartphones ont trouvé leur place dans les poches de la majeure partie du globe. Dans cette optique, nous avons analysé l'écosystème du suivi au sein des jeux Android. Notre analyse d'un jeu de données composé de 6 355 jeux gratuits et 396 jeux payants montre que les jeux payants ont effectivement moins de traceurs, mais ceux-ci restent malgré tout prévalents dans certains cas. Nous avons investigué le programme "Teachers Approved" de Google et montrons ainsi que le nombre de traceurs et de publicités est significativement réduit dans les jeux portant cette mention, soulignant l'importance de l'application des règles dans l'écosystème de la vie privée.

3. Finalement, nous introduisons une méthodologie de calcul de bout en bout de l'impact carbone de l'industrie de la publicité en ligne. Nous montrons que le fait de naviguer sur le Web en autorisant les traceurs et la publicité peut être responsable d'une augmentation du coût carbone de plus de 144%. Nos résultats montrent également que les régulations ont un impact positif au niveau environnemental : en instrumentant les bannières

à cookies, nous mettons en avant que le fait de refuser tous les cookies introduit une baisse significative de l'empreinte carbone de la session Web dans la majorité des cas.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

The Web is a constantly evolving organism. In 1990, Tim Berners-Lee was sharing the first Web page that showcased the hyperlink system on his NeXT computer at CERN. Today, the Web is home to countless websites and has a user base of over 5.3 billion users, that access their businesses, have social interactions, and create new content every day. All of what we currently witness on the Web has one starting point: Tim Berners-Lee's NeXT computer. It was the starting point for the creation of a constantly expanding infrastructure that currently handles an estimated 328.77 million terabytes each day. Nowadays, Web applications are being developed for multiple use and new devices are being connected to the Web every day. The smartphone boom in the early 2010s further pushed the Web into the very palms of Internet users, making it even more present in their lives: in their 2019 survey, Shimray *et al.* found out that almost 40% of their respondents spent one to three hours daily on their mobile phones [3], showcasing the prominence of the technology in our lives. The advent of modern smartphones has allowed democratized access to the Internet, with even more users being able to explore new social media platforms, play online video games, or simply gain access to new streams of information.

The success and exponential growth of the Web have also attracted many business actors that thrived through the monetization of their platform's content. Online advertising was born little after the creation of the first Web browsers: in 1994, the HotWired online magazine had allotted multiple square spaces for advertisers, which in turn helped finance their activities. Online advertising grew to become engrained in every facet of the Web: search engines, social media, shopping websites, and simple blogs all include advertising as a way to generate more revenue. Consequently, the advertising ecosystem evolved far from simple online banners to a complex and intermingled technical landscape, encompassing multiple techniques to precisely target and profile Internet users. Smartphones and connected devices in general have offered advertisers a simple and improved way to target users even more precisely: these devices are usually carried almost everywhere and are used to perform a wide range of personal activities. As such, mobile devices introduced an entirely novel perspective for the tracking industry. The information went from static and periodic to moving and constant. Consequently, the online tracking industry steadily evolved its techniques over the years to constantly improve its targeting accuracy on all devices. Today, two main forms of online tracking are prevalent on the Internet: stateful and stateless tracking.

Stateful tracking was enabled by the introduction of persistent storage under the mechanisms of browser cookies by the Netscape Navigator back in 1994 [4]. Their initial purpose was already related to tracking: website owners needed a simple way to identify recurring from one-time users. Today, there are multiple ways to take advantage of stateful tracking due to the diversity and multiplicity of browsers' APIs, and the simple process of saving tracking-related cookies for the benefit of a single website has evolved into complex and intermingled techniques that allow advertisers to precisely track users throughout multiple websites and browsing sessions, and even multiple devices, enabling them to build a precise advertising profile for each person they target. On the other hand, stateless tracking does not persist any information on the user's device. Mayer [5] first hypothesized in 2009 that the diversity of devices could be used to generate a fingerprint of Web users. Following in Mayer's footsteps, Eckersley [6], with the collaboration of the *Electronic Frontier Foundation* (EFF)[1], performed the first large-scale study of browser fingerprinting in 2010. The privacy implications of this technique have been further studied over the years in various works [7, 8, 9]. However, while browser fingerprinting has been established as an efficient means of identification in the short term, the instability of the used attributes significantly decreases its tracking efficiency for longer timespans.

The complexity of current ads, and ever more complex tracking techniques not only has a cost on the privacy of Internet users, but it also represents a significant share of the carbon emissions of the IT industry. According to the United States Environmental Protection Agency,[2] *global greenhouse gas* (GHG) emissions have increased by 43% between 1990 and 2015. *Data centers*, which host the majority of the content being accessed online, are estimated to account for 1% of global electricity demand. Therefore, it can be assumed that the constantly richer ads and compute-heavy tracking techniques introduce a major strain on the worldwide IT infrastructure, contributing to the rise of the IT industry's carbon impact.

Over the years, increased awareness about online tracking led to the development and democratization of privacy-preserving initiatives. Consequently, multiple tools have been developed that attempt to reduce the privacy risks on the Web: ad-blockers, such as AdBlock Plus[3] and uBlock Origin[4], have gained significant popularity over the years. Both rely on community-developed filter lists to block online trackers and advertisements from websites. In 2021, it was estimated that 42.7% of Internet users have relied on an ad-blocker [10]. Following public outcry, the browser industry has also introduced major initiatives, with Apple's Safari leading the way in 2017 with their Intelligent Tracking Protection (ITF) functionality, which focused on blocking third-party trackers. Mozilla enabled by default their *Enhanced Tracking Protection* (ETP) program in the Firefox browser in 2019. ETP's *standard* protection focuses on blocking third-party trackers, while their *strict* mode includes a browser fingerprinting protection, based on a list of known fingerprinting domains. Microsoft's Edge privacy initiative was also introduced the same year. *Microsoft Tracking*

[1] `https://www.eff.org/`
[2] `https://www.epa.gov/`
[3] `https://adblockplus.org/`
[4] `https://ublockorigin.com/`

Protection (MTP), behaves similarly in functionality to Mozilla's ETP. Google Chrome, which accounts for over 63% of the desktop's browser market, created the *SameSite* specification in 2016, which was originally created to restrain first-party cookies from being used by third parties. More recently, Google introduced the *Privacy Sandbox* to gradually replace the use of third-party cookies, while still maintaining the ability for the advertisers to target their ads.

This increased awareness of privacy on the Internet also extends to the mobile industry. Due to the growing scrutiny on privacy, Google gradually introduced a series of recommendations and guidelines on their Android ecosystem, that are targeted at offering more transparency for users and more customization options to give users more control over which data they choose to share with individual applications. Apple, on the other hand, set in motion a series of privacy-preserving changes in their iOS ecosystem, most notably with the *Apple Tracking Transparency* (ATT). This change imposed a strong requirement for apps on the App Store to explicitly request permission before performing any tracking. Finally, DNS-blocking is available on both platforms, either through the use of dedicated applications, such as AdGuard[5] or NextDNS[6], or through the configuration of a custom DNS server, with the help of open-source projects, such as Pi-Hole[7].

As the cat-and-mouse game goes on between advertisers and a part of the Web community, new tracking techniques are constantly being developed and countered. Novel tracking techniques may take advantage of the fast development of browser APIs to introduce novel attributes to enrich browser fingerprinting. While filter-list based ad-blockers remain efficient today, they require a substantial amount of collective contributions, as advertising domains are constantly tailored to evade the blocking rules. CNAME cloaking [11] allows advertisers to evade cookie-related defenses by disguising requests through DNS and subdomain manipulation, allowing them to set third-party cookies as first-party cookies. This constant adaptation raises the question of how many tracking techniques are already implemented but yet to be discovered by the Web community.

I personally witnessed the time during which the Web and the online advertising industry were gaining traction. My first encounter with online advertising was materialized by simple sparse banners, that quickly grew more invasive on the early websites I visited. Website owners attempted to generate revenue with advertising and tried to reserve a significant amount of space for ads on their pages. A few years later, pop-up ads became more prominent, to the point where it was hardly possible to navigate the Web without the annoyance of dozens of windows opening before or while navigating to the expected content. I have then witnessed the shift of the online advertising industry to targeted ads, thanks to more capable browsers and the advent of social media advertising. Finally, and most significantly, I witnessed the Web evolve from a mainly decentralized digital world to a centralized space dictated by the major Web actors.

This thesis is driven by my desire to shed light and improve the understanding of the online advertising ecosystem on both mobile and desktop, by investigat-

[5] `https://adguard.com/`
[6] `https://nextdns.io/`
[7] `https://pi-hole.net/`

ing and identifying novel tracking techniques and studying their stability and efficiency. Furthermore, the growing and increasingly visible menace of climate change motivated me to delve into energy efficiency by estimating the carbon impact of online advertising. Therefore, this thesis aims to answer the following loosely related research questions that attempt to improve our understanding of digital privacy, novel tracking techniques, and the underlying ecological impact of our decision to drive the Web through advertising:

- Can low-level browser APIs that rely on hardware acceleration be leveraged to improve and increase the reliability of browser fingerprinting?
- To what extent do free and paid mobile games differ in their inclusion of trackers and requests for permissions, and what are the privacy implications of these differences?
- What are the long-term environmental implications of the ever-increasing demand for data and extensive tracking algorithms of the online advertising industry?

1.2 Contributions

1.2.1 Exploring GPU fingerprinting as a stable and resilient tracking attribute

Browser fingerprinting has been gaining ground in the tracking ecosystem over the recent years. With Google announcing the end of third-party cookies for the end of 2024,[8] advertisers are tempted to look for tracking alternatives. Browser fingerprinting, while highly efficient at discriminating different devices from each other at a given time, suffers from the instability of many of its attributes, making it unsuitable for long-term tracking. Users might decide to use an external screen, install a new font, or update their browser. These changes lead to a new fingerprint being generated. Furthermore, the tracking accuracy of browser fingerprinting is affected by the existence of devices with similar hardware and software configurations. Through a set of heuristics and machine-learning based decisions, Vastel *et al.* [12] showed that the limitations of browser fingerprinting can be overcome. In *FP-Stalker*, Vastel *et al.* effectively tracked users for up to 2 months by evaluating changes in their fingerprints. In this work, we argue that the *Graphical Processing Unit* (GPU) of a device can be used to discriminate nominally identical devices.

We identify small differences among the *Execution Units* (EU) that constitute a modern GPU and fingerprint the GPU stack through *WebGL*. We find that our method can tell apart identical hardware configurations both in a controlled lab setup and in the wild. Our lab configuration is composed of 88 devices from nine distinct hardware classes, comprised of both desktops and mobile devices. We find that our technique can accurately identify identical devices with an accuracy of up to 95.8%. We further evaluate DrawnApart in the wild on a collection of 370,392 fingerprints from 2,550 unique devices through the AmIUnique platform. We find that a standalone evaluation of our technique introduces a significant improvement over the base rate. We implemented Vastel *et al.* [12]

[8] `https://developers.google.com/privacy-sandbox/blog/cookie-countdown-2023oct`

FP-Stalker and evaluated DrawnApart in conjuncture with other attributes and found that our technique extends the median tracking time by up to 66.66%.

1.2.2 Investigating the privacy impact of paid and free games on the Android ecosystem

Smartphones have reached their way into the pocket of over 80% of the world population, allowing people to access information at any time, work, or ultimately, play games independently of their location. Even before the advent of smartphones, mobile video games were prevalent and constituted a strong selling point for phones. Today, it is a huge industry worth over US$273 billion and over 2 billion players worldwide. The continuous growth of the mobile gaming market is pushing developers to follow multiple ways to monetize their games: in-app purchases, one-time fees, or subscription-based payments are among the currently adopted strategies. On the Play Store, most games are displayed as free, while the remaining are mostly cheaply priced, with most games costing under US$10. The targeted population for the mobile industry is wide, encompassing adults, teenagers, and even children. In the era of privacy awareness, it is important to understand the privacy implications that are introduced by the mobile gaming industry.

In our work, we built a pipeline to collect 6,355 free games and 396 paid games and used the *Exodus Privacy* tool to analyze each of these applications individually. For each game, we collected its metadata, which includes the game genre, the amount of downloads, or whether it contains in-app purchases. We find that paid games effectively contained fewer trackers and required fewer permissions from the users compared to free games, but there were instances where the amount of included trackers was still substantial. We identified multiple games that requested access to unneeded permissions, such as the camera or the location, that could be used for tracking purposes. Our analysis showed that the price of games is not correlated to the presence and amount of trackers they included, with many games priced on the cheaper side including fewer trackers than more expensive games. We further analyzed the type of trackers that games contained and found that advertisement-related trackers were prevalent in free games while analytics trackers dominated in paid games. Finally, we find that Google's *Teachers Approved* program effectively reduced the number of trackers and advertising in the games that included the mention, showcasing that strong and enforced guidelines have a positive impact on privacy.

1.2.3 Estimating the end-to-end carbon impact of online advertising

The Internet is constantly attracting new users. With over 5 billion users accessing the Web in various ways each day, the Internet is under constant pressure, with quintillions of bytes created daily. Advertisers seek to take advantage of this constant data flow and the ever-growing number of Internet users to distribute more ads and improve their tracking practices. This constant cycle of improvement puts a significant strain on the current infrastructures. In the current time and age, the stakes related to climate change are well-defined and affect every industry: data centers, which host most of the Internet content, are estimated to

be responsible for 1% of worldwide electricity usage. In 2008, Taylor *et al.* [13] estimated that every million ad impressions produced $676kgCO2_eq$. By 2023, tracking techniques heavily rely on complex algorithms and ads have become richer to capture the attention of users as quickly as possible, increasing the environmental impact of online advertising.

Thus, we propose AdCarbon, an end-to-end pipeline to perform fine-grain measurements of the carbon impact of the online advertising industry. In particular, the pipeline uses *Smartwatts* [14] and *Chrome's Devtools* to extract the carbon impact of each request individually, on the client side. It performs network measurements to estimate the share of the network in the total carbon emissions of a single request. And finally, through simulations, it estimates the *data center*'s environmental cost associated with the request. The pipeline is used to perform various crawls that span over 3 months. We find that browsing the Web without an ad-blocker leads to 144% more carbon emissions than having an ad-blocker. During our crawls, scripts were responsible for the vast majority of the carbon emissions, followed by images. We find that on the client side, the impact of the ad-bidding process is marginal and does not significantly increase the carbon cost of a browsing session. Finally, we find that accepting cookies on a website leads to a significant increase in the carbon cost of the session, leading to the conclusion that the GDPR's impact is globally positive both from a privacy standpoint and from an environmental standpoint. In this contribution, we provide recommendations for reducing the carbon impact of the online advertising industry.

1.3 Scientific publications

During my doctorate, I co-published the following papers:

[15] Tomer Laor[9], Naif Mehanna[9], Antonin Durey, Vitaly Dyadyuk, Pierre Laperdrix, Clémentine Maurice, Yossi Oren, Romain Rouvoy, Walter Rudametkin and Yuval Yarom. DRAWNAPART: A Device Identification Technique based on Remote GPU Fingerprinting. *Network and Distributed System Security Symposium*, Feb 2022, San Diego, United States.
Core rank: A / Google Scholar (Computer Security & Privacy): #6 / Acceptance Rate: 16.2%*

DrawnApart was awarded first place at the CSAW'22 MENA Applied Research competition and was cited in various newspaper outlets in multiple countries.[9][10][11]

[16] Pierre Laperdrix, Naif Mehanna, Antonin Durey, Walter Rudametkin. The Price to Play: a Privacy Analysis of Free and Paid Games in the Android Ecosystem. *ACM Web Conference 2022*, Apr 2022, Lyon, France.

[9] Co-first authors with equivalent contributions.

[9] https://www.forbes.com/sites/daveywinder/2022/02/05/the-next-graphics-card-crisis-could-be-the-most-worrying-yet/

[10] https://www.01net.com/actualites/ces-chercheurs-ont-trouve-comment-nous-identifier-sur-le-Web-grace-aux-cartes-graphiques-de-nos-pc-2054214.html

[11] https://www.heise.de/news/Browser-Fingerprinting-PCs-Smartphones-Co-lassen-sich-ueber-die-GPU-tracken-6345233.html

Core rank: A / Google Scholar (Databases & Information Systems): #1 / Acceptance Rate: 17.7%*

The Price to Play was selected for presentation at the 1st CNIL Privacy Day in Paris.

[17] Naif Mehanna and Walter Rudametkin. Caught in the Game: On the History and Evolution of Web Browser Gaming. *ACM Web Conference 2023*, Mar 2023. Austin, TX, United States.
Core rank: A / Google Scholar (Databases & Information Systems): #1 / Acceptance Rate: 19.2%*

[18] Naif Mehanna, Walter Rudametkin, Pierre Laperdrix, Antoine Vastel. Free Proxies Unmasked: A Vulnerability and Longitudinal Analysis of Free Proxy Services. MADWeb 2024 (colocated with NDSS), Mar 2022, San Diego, CA, United States.
Best paper award.

The following paper is currently under submission:

Naif Mehanna, Walter Rudametkin, Mohamed-Chakib Belgaïd, Pierre Laperdrix and Romain Rouvoy. Ad-Carbon: An End-to-End Analysis of the Carbon Footprint of Advertising on the Web. Submitted to IEEE Transactions on Sustainable Computing.

1.4 Scientific vulgarization

Throughout my academic journey, I presented my work at the following conferences, workshops, and events:

- Workshop presentation of DrawnApart at the LASER Workshop — San-Diego, CA, United-States (Remote) — March 2022
- Presentation of DrawnApart and "The Price to Play" at the *12ème Atelier sur la Protection de la Vie Privée* (APVP 2022) — Châtenay-sur-Seine, France — June 2022
- Presentation of "The Price to Play" to the CNIL Privacy Day — Paris, France — June 2022
- Presentation of AdCarbon to the French Ministry of Finances (PEReN team) — Paris, France — July 2022
- Presentation of DrawnApart and AdCarbon to the DiverSE team — Rennes, France — October 2022
- Thesis presentation to the LIFO Research team — Bourges, France — Nov. 2022
- Presentation of the "Caught in the Game" paper at TheWebConference 2023 — Austin, TX, United-States — April 2023
- Presentation of AdCarbon at the *13ème Atelier sur la Protection de la Vie Privée* (APVP 2023) — Arc-et-Senans, France — June 2023

- Presentation of AdCarbon to the Direction Générale des Entreprises (DGE) — Remote — Dec. 2023
- Presentation of *Free Proxies Unmasked* at the MADWeb workshop, colocated with NDSS 2024 — San-Diego, CA, United-States — March 2024

I have been an Artifact Evaluation reviewer for NDSS'24, and a reviewer for the IEEE Transactions of Games journal. I also sub-reviewed an estimated 10 papers for various venues including top-level conferences.

1.5 List of Tools and Prototypes

Our work on DrawnApart is publicly available in the following repository: `https://github.com/drawnapart/drawnapart` and a brief description is provided on the *AmIUnique* blog .[12] It includes the JavaScript and GLSL collection code for the online, offline, and GPU-based fingerprinting techniques. The machine-learning pipeline, along with various required datasets for reproducibility is also provided.

Our work on the privacy of games on the Android ecosystem can be found at `https://github.com/antonin-durey/the-price-to-play` and contains:

- The list of game IDs used for our study.
- The script to collect metadata from the Play Store.
- The scripts to collect APKs from AndroZoo and the Play Pass.

The artifacts of our AdCarbon contribution are available at `https://github.com/naifmeh/adcarbon`. It contains:

- The automatized pipeline for measuring the carbon impact of individual requests.
- The crawling scripts used to visit websites and instrument the measurements.

The dataset and testing tool set used in the *Free Proxies Unmasked* is also available at `https://github.com/naifmeh/free_proxies_unmasked`.

1.6 Outline

This thesis is organized as follows.

Chapter 2 introduces the context of this thesis. I first present an overview of the past and current state of the online advertising ecosystem and discuss what led the online advertising industry to become today's Web behemoth. I then define the important notions of this thesis, by going through global mechanisms of online tracking and online advertising. Finally, I introduce the context of carbon measurements on the Internet and how it can be applied to measure the carbon emissions of online advertising.

[12] `https://blog.amiunique.org/an-explicative-article-on-drawnapart-a-gpu-fingerprinting-technique/`

Chapter 3 presents my study on GPU fingerprinting. The state-of-the-artpresents no simple methods to distinguish between devices with identical software and hardware configurations. In this work, along with my co-authors, I introduce DrawnApart, our GPU fingerprinting technique, which is capable of distinguishing between identical devices by exploiting small differences in the execution speed of elementary operations on the GPU. I present the evaluation of our method in controlled settings, showing that DrawnApart is capable of distinguishing between devices issued from the same production line. In this chapter, a one-shot pipeline that leverages DrawnApart and can be used in the wild, with little overhead, is also introduced. The performances of the one-shot pipeline are measured against a crowd-sourced real-world dataset obtained through AmIUnique.

Chapter 4 investigates the privacy implications of paying for games, compared to playing their free counterpart on the Android ecosystem. To this end, I present in this chapter our collection methodology, which leverages alternative stores to bypass Android's PlayStore restrictions while still retrieving the official application's metadata. Our pipeline then extracts analytical insights using the Exodus module. Later on, through our analysis, we show the prevalence of trackers and the relevance of requested permissions in both types of applications and measure their prominence per game type. We look at whether in-game purchases and the initial game's price have an impact on the presence of trackers. Finally, we provide an extensive analysis of one category of games that are specifically tailored for kids.

Chapter 5 explores the carbon impact of the online advertising ecosystem. No existing work has attempted to perform a fine-grain measurement of the carbon impact of individual advertising requests. In my work, titled *AdCarbon*, I present a measurement pipeline capable of estimating the carbon impact of individual network requests through a combination of real-world measurements and estimations that take into consideration the client's device, the network, and the data centers' costs. Our overall theoretical model is also defined. In the following sections, I introduce the results obtained through the measurement pipeline and outline the carbon impact per country, per data type, and per website category. Next, I look at how the cookie banners impact the overall carbon cost of the webpage and whether header-bidding introduces a significant carbon overhead when employed.

Finally, in Chapter 6, I conclude this thesis by summarizing my contributions, proposing future works, and discussing the current and potential future trends of a more privacy-preserving online advertising ecosystem.

Chapter 2

Background & Context

2.1 Evolution of the Internet

The Internet as we know it today is the result of a steady technological evolution that was initiated in 1969 with the creation and first use of the ARPANET. Similarly to the reasons we use the Internet today, the ARPANET was born from the desire to share information over great distances, in a decentralized manner. At its beginnings, the ARPANET only hosted four nodes on US soil. But the reliability and ease of communication that the system introduced led more computers to join the network, and by 1984, over 1000 hosts were connected. What stemmed from the desire to share scientific research seamlessly between computers became the basis of what is known today as the Internet: ARPANET was the pioneer of packet-based communication and the TCP/IP protocol suite, which are the foundation of today's Internet. ARPANET's nodes were not only used for scientific research but also for games and direct messaging. As the popularity of the network grew, it quickly became apparent that a much larger network was needed to accommodate the exponential growth of users, leading to the ARPANET being decommissioned in 1990, leading to the creation of *the Internet*. Even before it was decommissioned, the success of the network led multiple entities to develop their own concurrent network: an example of such a network is the privately owned *Transpac* network [19], created in 1978, which led to the ephemeral success of the French Minitel [20]. However, most concurrent networks were ultimately interrupted by the quick adoption of the Internet.

As the ARPANET was being decommissioned, the academic community was still desperate for a simple way to share information universally. In 1989, Tim Berners-Lee came up with a proposal showcasing his *hypertext* project, which described the *Web* as the collection of multiple documents that were interconnected through *hyperlinks*. This proposal was the basis for what is known today as the *World Wide Web* (WWW). By the end of 1990, Tim Berners-Lee implemented the first Web server on his NeXT computer. This first Web page was coded using a basic version of the *Hypertext Markup Language*, which is commonly known today as HTML, and could be accessed through the *World Wide Web* browser, which was initially limited to NeXT computers. The success of Berners-Lee's vision led to the quick adoption of the Web, and a year later, the *Mosaic* browser[1] was created, showcasing color graphics in addition to regular text. In 1993, the Mosaic browser was used by over a million people. It evolved to become known as the highly successful *Netscape Navigator*. The *Netscape Navigator* introduced many of the features that can be found in

[1] `https://archive.org/details/mosaic-ncsa-evolt_browsers`

today's browsers: the *Cascading Style Sheets*, which allowed publishers to alter the looks of their pages, and most notably, the *Javascript* scripting language, which introduced interactivity in Web pages. Due to the commercial success of the Netscape Navigator, Microsoft decided to introduce their own navigator with the creation of *Internet Explorer*, leading to the *first browser war*.

The browser wars that involved Internet Explorer and the Netscape Navigator along with its successor, Mozilla Firefox, was a period of rapid development in browsers, driven by a fight for the most market share. Browsers went from a simple means of accessing Web pages, to a central tool that could encompass all the user's needs: from the simple task of accessing a static Web page to playing games or creating 3D objects, browser developers introduced many new APIs that enabled Web page publishers to attract users to their websites. The Web was now home to rich applications, enabled by the growing capabilities of *Javascript* and the plugin system that allowed browsers to execute arbitrary programs if they provided a compatible plugin. Java applets and the Flash plugin were two significant examples of such plugins. The Web had evolved and with it, various new threats were put to light.

This rapid development of browsers was often done without much regard to security: as such, many new APIs were introduced with major security flaws. In 1997, Dean *et al.* [21] were already hinting toward major security flaws that were enabled by the introduction of Java applets in the browser. De Paoli *et al.* [22] listed various attacks that were possible in browsers in 1998: more notably, they also outlined the risks of specific features, such as the recent introduction of cookies and the extended system privileges of applets, regarding user privacy. As pointed out by De Paoli's study, browsers at the time already included tools aimed at protecting the privacy of users on the Internet.. Such tools included the ability to disable the execution of Java and Javascript or alert when a Web page attempted to save a cookie. Interestingly, Internet Explorer 4.0 introduced the ancestor of the Site Isolation feature,[2] by isolating websites into four zones which included different levels of security and limitations. In 2000, Felten *et al.* [23] describe various timing attacks and privacy risks on Web browsers. More precisely, Felten *et al.* studied the role of caching in timing attacks and deplored that most of these can hardly be mitigated by browsers, because they are mainly due to basic properties of Web browsers. They further note that such attacks greatly hindered user privacy on the Web, by allowing attackers to learn about users' browsing history through the measurement of the time it took to access a cached or non-cached resource. The authors also note that caching attacks are not limited to cached resources but can also be extended to DNS requests, by measuring the time it takes to resolve a cached and non-cached domain name, and cached cookies, which are an invasive form of web cookies that are stored in the user's cache without the user's knowledge.

By the early 2010s, only a few years after its introduction, Google Chrome overtook Internet Explorer and then Mozilla Firefox as the most popular browser, which still holds true to this day. As of August 2023, Chrome holds over 63% of the market share. This domination is arguably the result of the improvements that were introduced by the Chromium engine, which introduces significant per-

[2] `https://www.chromium.org/Home/chromium-security/site-isolation/`

formance (mainly because of its V8 JavaScript engine[3]) and security gains over its predecessors [24, 25]. This widespread adoption contributed to an increased focus on the security of the newly introduced features.

In the meantime, the Internet, which was initially limited to desktop computers, witnessed the introduction and exponential growth of the mobile industry. The first truly mobile and commercially successful phone was presented to the public in 1973 by Motorola. The phone delivered the usual phone features: calling and receiving calls, but with the distinction that phone calls could now be taken anywhere and were not limited by the location of the device. But it wasn't until 2007 with the unveiling of the first *iPhone* by Steve Jobs that smartphone usage became common. Prior to that, various attempts to introduce Internet-enabled devices to the public varied in success depending on the device: Blackberry, for example, pushed their Web-enabled device in 2002 and managed to win over a business-oriented public with promises to boost productivity by using their mobile phones which provided facilitated usage of emails, notes, faxing and text messaging [26]. The iPhone was the first smartphone that was directly targeted at consumers with its ease of use, while also providing an advanced infrastructure for software developers, which fueled the boom of dedicated applications [27].

Motivated by the great success and promises of the new generation of smartphones, the Android ecosystem emerged as a strong competitor to Apple's iOS in 2008. As opposed to iOS, Android was set to be open-source and could be used and modified by any smartphone constructor. By 2015, over $24,000$ different models of Android-based smartphones were available to the public, showcasing the quick adoption and success of the operating system. Following in Apple's footsteps, Android also introduced the Android SDK, a suite of tools for software developers wishing to develop applications for the operating system: convinced by the operating system's adoption rate, the number of applications grew rapidly, reaching over $3,550,000$ applications in 2022. On the other side, the App Store accounted for over $1,642,000$ applications in the same year. This lower number is at least partially explained by the stricter access conditions of the App Store, which requires developers to follow a strictly enforced set of guidelines [28] and has a higher cost [29], compared to the Play Store [30]. The success of both ecosystems can easily be shown through their adoption numbers: by 2023, Apple and Android both concentrated over 99% [31] of the mobile market share.

As specifications continue to improve, the bridge between desktop computers and smartphones is shrinking: smartphones represent a cheaper alternative, with the availability of a wide range of financially accessible models, that can perform most of what could only be performed on a desktop computer. Applications have adapted to offer the same, or even better services for their mobile versions than their desktop or Web versions. For example, the world-renowned image editor, *Adobe Photoshop*, previously available only on computers, was introduced to the mobile ecosystem in 2016. Some applications are even entirely restricted to the mobile ecosystem. Online shoppers are also favoring mobile platforms for their purchases: in 2022, Faverio *et al.* [32] found that about three-quarters of US citizens report buying things online through a smartphone, and this trend is particularly pronounced among younger people.

[3] `https://v8.dev/`

Silver *et al.* [33], in a study conducted in 2019, found that smartphone ownership is high even in emerging countries with low income. They found that smartphone ownership reached 97% among adults in Vietnam, with a median percentage of 89% among the 11 countries concerned by the study. Similarly to Faverio *et al.* 's findings, the younger generation was found to be more prone to possess a smartphone than older people. These figures are particularly noteworthy given the fact that in all the studied countries but one, personal computers were owned by less than half the population.

In conclusion, from the start of the ARPANET to the widespread use of browsers such as Google Chrome and the surge in smartphone ownership, multiple transformations have shaped the way we use the Internet. These changes have effectively impacted the way we communicate, access information, and go about our daily routines. However, as our lives increasingly rely on digital platforms, a special emphasis should be placed on understanding and acknowledging the privacy issues on the Web.

2.2 Online advertising

Early in the Internet's lifespan, online advertising was mostly frowned upon, both by users and infrastructure pioneers: the ARPANET, among its acceptable use policies, banned the presence of commercial activities by for-profit institutions.[4] These guidelines, while not entirely preventing various forms of advertising on these networks, greatly limited and postponed the introduction of online advertising to the Internet. However, some website owners and businesses argued that monetization of the Internet was necessary for its growth and continuity. Among those websites, the *HotWired* blog was the first to allocate space for what are deemed to be the first online ads. The first advertising slot, a 460x60 pixels rectangle on top of the page, was referred to as an *"ad banner"* and was sold to *AT&T* for 3 months at an upfront fee of US$30,000. The tremendous success of the ad, which showcased a click-through rate of over 42% [34], initiated the exponential growth of online advertising. Many websites started adopting the same business model, offering multiple slots on their pages to advertisers, in a bid to finance their activities. Early Internet users were reticent to the presence of online advertising on the Web for many reasons that remain closely aligned to today's user sentiments toward ads: in the era of dial-up modems, it was proven that graphical ads slowed down websites. This sentiment was further exacerbated by the fact that many websites placed an extensive amount of ads, drowning their actual content and confusing users [35, 36].

After these first experimentations, advertisers were quick to look for other means of distributing advertisements to the online community. The exponential growth of the Web in the 1990s saw the rise in popularity of search engines and with it the beginning of search-based advertising. Websites were looking for a way to advertise their presence and the early search engines provided a way to promote them for a fee. *Goto.fr* was one of those search engines and allowed websites

[4] `https://www.ccexpert.us/network-mask/origins-and-recent-history-of-the-internet.html`

to bid for a better placement when users were searching for an associated keyword [34]. This advertising strategy pioneered the *Pay-Per-Click* model, which allowed advertisers to only pay when a click ratio had been met. However, this model made search engines unreliable and led to poor results being displayed, as websites that could pay more money were always guaranteed to be on top, regardless of their relevance.

The introduction of *browser cookies* by the Netscape Navigator in 1994 brought significant promises to the booming online advertising industry. Cookies were originally created with traceability in mind: before their introduction, website owners had no way to differentiate returning users from new users, and therefore, no possibility to customize their experience. This limitation was particularly problematic for online stores, which could not keep track of items in a user's shopping cart across multiple sessions without requiring explicit user login.

The creation of cookies was meant to allow websites to improve the user experience by making them traceable across a single website. The public was quick to note the privacy issues that came with this new feature: in 1996, the Financial Times published an article titled "This Bug in Your PC is a Smart Cookie", [37] raising awareness about the privacy risks of cookies. However, in the early years of the Web, privacy was relegated to second place, to the benefit of quick innovation and improved business models. The online advertising industry quickly realized that cookies had the potential to be used to precisely target users across various websites. The birth of *ad-exchanges* paved the way, while first-party cookies mostly allowed website owners to improve the usability of their website, third-party cookies were exploited by advertisers to track users across browsing sessions. Advertisers could use the reach of their ad exchange to build a profile on the users based on their activity over the Web. Consequently, the ads that they displayed were precisely tailored to the needs and wants of the tracked user, leading to bigger benefits for the advertisers.

During the 2010s, the advertising industry spread in many directions: video advertising took off as *Youtube*[5] was created, social-media advertising showed great promises thanks to the growing popularity of social networking and mobile advertising started to appear as smartphones were becoming popular. Social media advertising turned out to be one of the most impactful directions taken by the advertising industry, generating a significant share of the online advertising revenue[6]: advertisers now had a way to connect and interact with groups of persons, giving brands the possibility to have an identity on the social network. Furthermore, due to the success of social media, existing platforms could acquire a significant amount of data, which they could leverage to build improved and more precise profiling and targeting algorithms. *Meta*[7] (previously *Facebook*), thanks to its success and popularity, became the most popular platform for hyper-targeted advertisement [38], which is the use of detailed user data and automation to display highly targeted and personalized messages ads. Thanks to the number of interactions happening daily on its platform, Meta became a leading actor in hyper-targeted advertising [38], arguing that users

[5] `https://youtube.com/`

[6] `https://www.iab.com/insights/internet-advertising-revenue-report-full-year-2022/`

[7] `https://facebook.com/`

were more prone to respond to *relevant ads* rather than when displaying a large number of ads. Today, Meta remains one of the major actors in the online advertising ecosystem with tracking technologies, such as the Meta Pixel and the Meta Conversions API, allowing them to identify users even outside their platform [39].

Growing fears for privacy and over-exposure to ads led the online community to look for ways to limit the risks and disturbances associated with online ads. From these initiatives, ad-blockers were born. In 1996, the first ad-blocker, named *Internet Fast Forward*, was published as a paid plugin but was quickly discontinued due to fears of potential lawsuits. In 2003, the popular *Adblock* was released and later led to the creation of *Adblock Plus*.[8] Noting the growing popularity of the ad-blocker, Google initiated a legal battle against *AdBlock Plus* in 2014, which ultimately resulted in the German Supreme Court ruling in favor of *AdBlock Plus*, further motivating different actors on the Web to join the ad-blocking train. In a recent study, Yan *et al.* [40] estimates that ad-blockers are being adopted by up to 43.2% of users, showcasing the growing defiance toward ads and the increased consciousness on Web privacy [41]. However, conventional list-based ad-blockers remain mainly prevalent on desktop browsers, as not many of their mobile counterpart include support for browser extensions in their mobile version. Furthermore, the majority of the most popular websites provide an app in both the Play Store and the App Stores, leading many users to use the application rather than the website version when on mobile, preventing the usage of conventional ad-blockers, and allowing the app developers to include tracking and advertising directly in the app. This offers significant incentives for advertisers to invest in mobile advertising: by 2024, it is expected that advertisement spending will increase to almost US$400 Billion, more than doubling compared to 2018 [42].

Today, online advertising as an industry is worth over US$638 billions [43] and spans across the entirety of the Web: from video-based ads to search advertising, online advertising's spending is expected to keep growing in the future years as more and more users get access to the Internet. Improved algorithms also lead advertisers to create better profile users and provide increasingly more targeted ads, that are tailored to the users' likes and wants, which in turn increases the advertising revenue.

2.3 Environmental impact of the Internet

Climate change is a defining challenge of our current era, dictating and weighing over most of our actions, both on the personal and global levels. More notably, the last decade has witnessed the consequences of global warming, with an increase in extreme weather phenomena. Climate change is not new, it first occurred in the minds of scientists over a century ago that our activities were causing a steady change in the climate of our planet. In reality, the change in our climate started back in the 18th century, with the advent of the Industrial Revolution. Human activities were and are still largely based on fossil fuels, which in turn increase the concentration of gases in the atmosphere at an unprecedented rate in more than $10,000$ years [44], such as carbon dioxide, methane,

[8] `https://adblockplus.org/`

or nitrous oxide. Starting from the 1950s, advances in computers and a better understanding of climate science allowed scientists to report and warn about the consequences of human activities. However, awareness remained low among the public and authorities. Global consciousness only happened relatively recently: in 1985, it was found that the protective ozone layer in Earth's atmosphere was reporting abnormally low levels of ozone over the South Pole [45]. Media coverage and easier access to information at that time led to a global shockwave that contributed to helping the public acknowledge the inherent risks of climate change. Following this acquired consciousness, multiple world agencies started cooperating to counteract the effects of climate change and in 1988, the *Intergovernmental Panel on Climate Change* (IPCC) was founded with the objective of providing research on climate change. The IPCC reports consistently proved that the planet is warming due to the release of greenhouse gases by human activities. These reports led to the current actions that are being taken to this day: in 2015, based on the reports, 195 countries agreed to limit global warming to less than 2 °C above pre-industrial levels. Governments also started acting against climate change by introducing a series of measures dedicated to reducing the use of fossil fuels following scientific recommendations, by promoting the use of sustainable energies, and by local awareness campaigns.

As a result of the global consciousness, the literature started employing the term *carbon footprint* as a measurement of the greenhouse gas emissions that are emitted by an activity. The search for solutions to limit the environmental impact of the world requires a better understanding of the impact of our individual activities, which might encompass the simple activities of cooking, browsing the internet, or reading emails. The *carbon footprint* measurements allow us to have a semantically simple estimation of how much impact this activity has on the environment. As opposed to what its name potentially indicates, the measurement accounts for the release of a number of different gases with a high climate warming potential, which includes methane, nitrous oxide, fluorinated gases, or carbon dioxide. It is expressed as a measure of the *carbon dioxyde equivalent (CO_2eq)* [46, 47, 48]. While the carbon footprint presents the advantage of offering one centralized metric to estimate the GHG emissions of an activity, various means of calculating it are provided in the literature [49, 50, 51], offering little coherence to the metric. As shown by various studies [48, 52], disagreements exist in the selection of gases, and the order of emissions that are to be included in the carbon footprint calculations. Different methodological issues also affect the calculation of the metric, as several methods have been established by national and international standards, but recent scrutiny has motivated efforts to provide a standardized method of calculation [46]. Today, the carbon footprint has been widely adopted by the scientific community as the way to measure the carbon impact of human activities. Multiple online tools exist to estimate the carbon footprint of a wide range of activities both at the industrial and individual levels [53, 54].

As *Information and Communication Technologies* (ICT) are concentrating an increasing amount of individual's time [55, 56, 57], many of the carbon footprint online calculators focus on Internet activities. At the same time, multiple studies have also shed light on the carbon footprint of online activities, such as the development and sales of online games [58], social media [59] or movie streaming [60], among others. The growing number of users of the Internet, and

ICT in general, calls for a better understanding of the carbon impact of digital individual activities: connected-device ownership is reaching all-time highs year over year and the increasingly connected world we live in is promising no decrease in this trend. Personal sensitization to the carbon impact of these appliances can be passed through two points: the life-cycle assessment of the appliance, which describes the environmental impact of a product from its conception to the day it is discarded, and the carbon footprint of the appliance's usage [61]. While the life-cycle assessment might lead users to limit purchases of highly emitting products, ownership of some connected devices, such as a smartphone or a personal computer, remains necessary in the current world. A study by Belkhir *et al.* [62] performed in 2018 expects the share of greenhouse gases emitted by the ICT industry to increase by 14% over the 2016 greenhouse gases emissions, mainly due to the continuous growth of smartphone usage, data-centers and communication networks. Their findings confirm the results from Biczok *et al.* 's [63] study, which forecasted a significant growth of the ICT carbon footprint, from 86 $MtCO_2e$ in 2007 to 235 $MtCO_2e$ by 2020.

Due to the broadness of the ICT sector, it is difficult to go above the estimations and provide more fine-grain measurements of its carbon footprint. As such, the carbon impact of different sectors of the Web has been under-explored in the literature. More specifically, one of the biggest share of Web traffic is linked to online advertising [64, 65, 66]. Englehardt *et al.* [65] in their measurement of online tracking on a dataset of 1-million websites, have shown that over 35% of performed requests were related to online advertising. Vallina-Rodriguez [66] found that about 18% of web traffic was specifically related to mobile advertising, back in 2012. This number is expected to be significantly higher as the adoption of smartphones has grown in the subsequent years.By exploring the carbon footprint and bridging the existing gap in the literature, it is possible to motivate a collective effort to mitigate the mounting environmental cost of the Web.

2.4 Online Tracking & Advertising

2.4.1 Online Tracking

Stateful Tracking

To properly function, browsers provided websites with multiple storage options. Authentication identifiers can be persisted thanks to browser cookies. Web resources can load faster if they are saved in a browser cache. However, advertisers and website developers were quick to notice that such tools could be used to track users. Today, stateful tracking is arguably the most effective and the most used form of tracking on the Web.

Cookies are the most commonly used technology that permits user tracking. They are textual data that are stored in the client's device. Two types of cookies exist: *1st-party* cookies, which are cookies set by the visited website. And *3rd-party* cookies, which are set by external entities that are included on the website. Third-parties might represent JavaScript libraries or font providers, analytics companies, or advertising companies. Initially, *1st-party* cookies were used to reidentify users upon return to the website, while *3rd-party* cookies were

employed to track users across various websites on which the third party is implanted. In 2016, Englehardt *et al.* [65] performed a large-scale measurement of the prevalence of online tracking: their findings show that a limited set of companies are prevalent on the vast majority of the websites they visited. Google, for instance, is present on almost 70% of the websites. By being present on a vast amount of websites, entities such as Google and Meta are able to build a browsing history for each tracked user, which in turn permits the construction of precise user profiles. Third-party tracking has been around since the early days of the Internet and has therefore been extensively covered [4, 65, 67, 68, 69, 70].

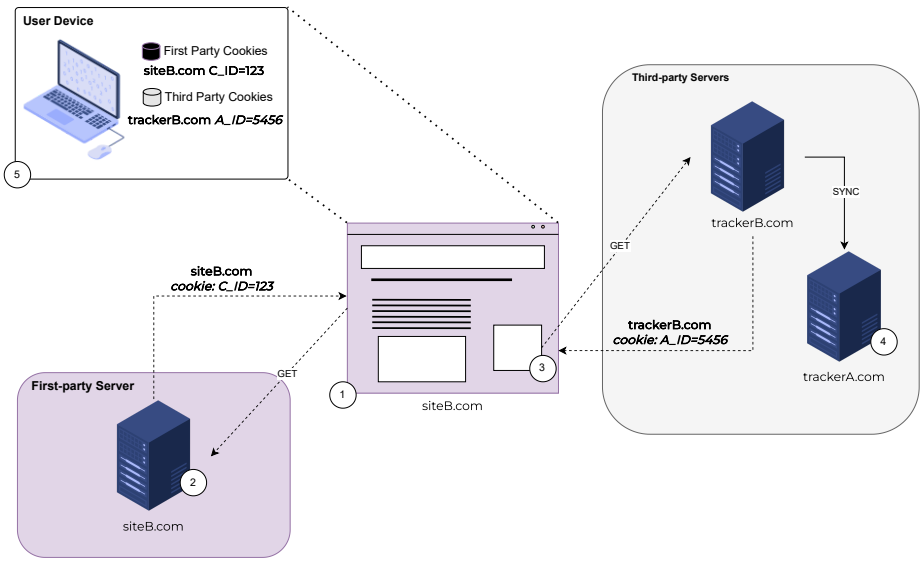


Figure 2.1: Overview of the cookie syncing process. (1) The user requests the siteB.com website, which performs first-party requests to its backend and gets a first-party cookie in response (2). The trackerB.com third-party initiates its GET request and includes a third-party cookie in its response (3). (4) TrackerB.com communicates to trackerA.com that the user visited siteB.com and transmits its ID. (5) Both first-party and third-party cookies are saved in the user's device.

For entities that have a more reduced presence on the Web, information sharing has been employed to build more precise user profiles. In fact, different third-parties might provide access to their cookie IDs in order to re-identify users on the websites they cover. As an attempt to limit this practice and the privacy risks it introduces, the *Same-Origin Policy (SOP)* [71] was introduced to reduce the amount of shared information by limiting access to cookies to the same origin as the creator. However, the advertising industry strongly relied on information sharing: one of its most notable uses was dedicated to Real-Time Bidding (section 2.4.2). To circumvent the limitations introduced by the SOP, the advertisers initiated the *cookie syncing* practice [72]. To illustrate the working principles of *cookie syncing*, assume a user visits *siteA.com*, which includes *trackerA* and *trackerB*. Both trackers create their own cookies with their own tracking ID. The user then goes to *siteB.com*, in which only *trackerB* is present. As part of agreements, *trackerB* answers to its third-party request

with an *HTTP REDIRECT* to a URL provided by *trackerA*, with information stating that the user visited *siteB.com*. The cookie syncing mechanism happening in *siteB.com* is illustrated in Figure 2.1. In their study, Papadopoulos *et al.* [72] found that 97% of users, including mobile users, were subject to cookie syncing, corroborating the results by Englehardt *et al.* [65], with an average of one synchronization for each 68 HTTP request. Worryingly, they found that through the cookie syncing process, not only are tracking ID shared, but various Personally Identifiable Information (PII) are also leaked, such as the user's city, phone number, or gender. Beyond the privacy risks that it introduces, cookie syncing also hinders existing regulations (section 2.4.3), making it significantly harder for users to exercise their GDPR rights.

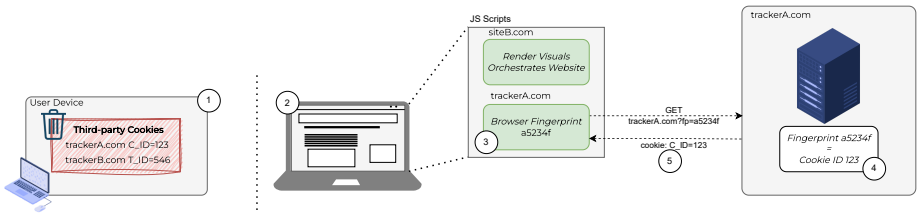


Figure 2.2: Overview of the cookie respawning process through browser fingerprinting, as described in [1]: (1) The user proceeds to delete their existing cookies; (2) The user accesses *siteB.com*, which contains a script collecting the device's browser fingerprint (3); (4) The browser fingerprint is transmitted to *trackerA.com*, which is then matched with the previously deleted cookie ID; (5) *trackerA.com* responds with the respawned cookie.

Cookie respawning, is another tracking mechanism that has been introduced to circumvent existing tracking protections. As presented in Section 2.4.3, various measures exist to limit stateful tracking, by removing them or preventing them from being set. Through cookie respawning, the same information is stored twice: once as a regular cookie and again using another type of storage. When the user returns to the website and is identified, the third-party attempts to access the cookie: if it doesn't exist anymore, it is respawned through the secondary storage. Multiple techniques can be used to respawn cookies. Soltani *et al.* [73] first coined the term in 2010 and identified Flash cookies being used to respawn traditional HTTP Cookies, since Flash cookies could be set with an indefinite expiration. Later in 2011, Ayenson *et al.* [74] make similar observations involving HTML5 Web Storage[9] and ETags[10]. Finally, Fouad *et al.* [1] performed a large-scale study and show that $1,150$ domains of the $30,000$ crawled domains use browser fingerprints to respawn cookies, and outline how this practice might go against the GDPR and the ePrivacy Directive. The cookie respawning process through browser fingerprinting is depicted in Figure 2.2.

Due to their abuse and highly invasive tracking, plans to end third-parties cookies have been announced by Google and are set to take place in 2024 [75]. With this in mind, advertisers have been shown to establish partnerships with first-parties in order to circumvent the upcoming end-of-life of third-party cookies. Demir *et al.* [76] have shown that over 85% of the $15,000$ websites they analyzed

[9] `https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API`
[10] `https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/ETag`

performed first-party tracking and shared this information with third-parties. In their work, they show that while it is not new, this technique has grown significantly since 2019. Most notably, their work shed light on the main entities misusing first-party cookies for tracking: Google, which is responsible for the end of third-party cookies, is the biggest user of this technique, followed by Meta and Adobe. Chen *et al.* [77] attempted a similar study and found that over 97% of the websites they visit include first-party cookies that are set through third-party JavaScript code included in the loaded page.

Stateless Tracking

The public's rising awareness of privacy issues on the Web and Google's announcement of the end of third-parties cookies [78] are leading advertisers to look for other techniques to track users. Stateless tracking, through browser fingerprinting, allows advertisers to perform their tracking without leaving a trace on the users' browsers.

Throughout the evolution of the Web, browsers have continuously integrated new and more advanced features in the form of browser APIs. Such APIs allow webpages to display 3D content, through the WebGL and the Canvas API, sound through the WebAudio API, and even Virtual Reality content through the WebXR API. This high diversity of APIs, many of them low-level hardware accelerated, has been introduced as an attempt to shape and provide a stable cross-platform user experience. But it didn't take long for the web community and advertisers to notice that this diversity could be taken advantage of. In his senior thesis in 2009, Mayer [5] emitted the idea that browsing environments could be exploited to provide identifying information on the users. Most importantly, Mayer estimated that the operating system, the browser configuration, and the hardware could lead to the browser presenting unique features. To test his hypothesis, Mayer collected three attributes (*navigator.plugins*, *navigator.screen*, *navigator.mimeTypes*) from 1328 clients and showed that 1278 could be uniquely identified.

Since Mayer, browser fingerprinting has been extensively studied: it consists of a set of information represented by software and hardware attributes that are obtained through the browser. A subset of those attributes is depicted in Table 2.1. These attributes are collected through the execution of scripts included in webpages: some attributes can be obtained by parsing HTTP headers, while most are obtained through JavaScript calls to browser APIs. When scrutinized individually, each attribute does not represent is not particularly unique. However, when a subsequent number of attributes are combined, they act as a *fingerprint* that provides a potential identifier of the user's device. After Mayer, Eckersley [6] was the first to perform a large-scale measurement of the prevalence of browser fingerprinting on the Web. Through his study, Eckersley analyzed $470,000$ *fingerprints* collected by their *PANOPTICLICK* platform and found that over 94% of browsers were unique. While they emphasize the tracking potential of browser fingerprints, Eckersley states that the fingerprints showed low stability over time.

Further studies investigated browser fingerprinting following Eckersley's publication: in 2013, Nikiforakis *et al.* [79] assessed the tracking potential of browser

fingerprints and showed that existing defense mechanisms, such as *attribute spoofing*, potentially make the user more prone to identification. This was confirmed by Vastel et al. in 2018 where every tool they tested to spoof fingerprints was easily detectable and some were severely counterproductive [12]. In the same work, Vastel *et al.* show that many attributes are unstable but there is sufficient stability and entropy to track a large set of users over time. In their dataset, about 26% of users could be tracked over 100 days exclusively using their browser fingerprint. When combining tracking techniques we can only expect the situation to be much worse for privacy. Also in 2013, Acar *et al.* [80] presented *FPDetective*, a framework to detect scripts performing browser fingerprinting. In 2014, Acar *et al.* [69] measured the adoption of *canvas fingerprinting* on $100,000$ websites and found $5,542$ of them performing canvas fingerprinting. In 2016, Laperdrix *et al.* [7] analyzed over $118,900$ fingerprints composed of 17 recent attributes and showed that the advances in web browser technologies provided highly unique attributes that can be used to create unique fingerprints. Al-Fannah *et al.* [81] crawled $10,000$ websites in 2018 as an attempt to assess the usage of browser fingerprinting in the wild. Though their definition of fingerprinting is broader than any previous work, they measure over $6,876$ websites performing browser fingerprinting, with the vast majority of fingerprinting being initiated by third-parties. For a complete overview of the state-of-the-art, in 2020, Laperdrix *et al.* [8] surveyed the existing literature on browser fingerprinting and present the existing countermeasures.

In 2021, Iqbal *et al.* [82] developed FP-Inspector, a tool that uses a combination of static and dynamic analysis to detect fingerprinting scripts. FP-Inspector has shown that as of 2021, browser fingerprinting was prevalent on over 10% of the top $100,000$ websites. Fietkau *et al.* [83] identifies the same trend using the FPMON tool, and find that over 19% of the top $10,000$ websites display fingerprinting patterns. In the same year, Sjosten *et al.* [84] introduced EssentialFP, with the same purpose as FP-Inspector. EssentialFP make use of dynamic analysis of the Javascript scripts along with an analysis of the requested network endpoints to identify scripts that are used to fingerprint users. More recently, Su *et al.* [85] proposed an automatic tool for discovering browser APIs that are being leveraged for constructing fingerprints. Through their work, they identified 161 APIs that are discovered uniquely through their tool and find that the fingerprinting ecosystem is fast-paced, with 18 APIs used in fingerprinting being identified only 11 months after their first measurement.

Although all the previous studies assess the tracking capabilities of browser fingerprinting, most of them fail to mention the main limitation of the stateless technique. The user environment is a constantly changing piece of information: users might update their screen, add new browser extensions, or change their browser language. Each of these changes introduces an update in their browser fingerprint, hindering long-term tracking. However, Vastel *et al.* [12] show through FP-Stalker that by employing a set of heuristics and automated learning, it is possible to overcome this limitation. In their work, they show that browsers can be tracked for 54.48 days on average, and 26% of browsers can be tracked for more than 100 days using only their browser fingerprint (i.e., without cookies or IP addresses). Another possibility to increase the stability of browser fingerprinting is to exploit stable and unique hardware attributes. Existing works have explored various attributes based on hardware features: in 2012,

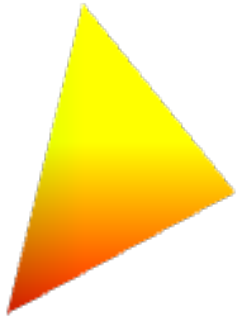
Mowery *et al.* [86] showed how the Canvas API could be exploited to produce attributes with high entropy. Their technique is now commonly referred to as *canvas fingerprinting*. In 2019, Queiroz *et al.* [87] exploited the Web Audio API to generate an attribute capable of identifying the device type, the web browser version, and the rendering engine. Tranpert *et al.* [88] attempts to fingerprint the device's CPU and propose six benchmarks, implemented in Javascript and WebAssembly, that exploit side-channel information to infer CPU properties, such as cache sizes, the number of CPU cores, and single- and multi-threaded performance characteristics. Through a study involving 834 participants with 297 different CPU models, their benchmarks achieve accuracies of up to 100% for inferring the mentioned microarchitectural properties.

Despite being mainly focused on the tracking abilities of browser fingerprinting, its usages are not limited to the advertising ecosystem: browser fingerprinting has been shown to be capable of being used to improve the user experience by providing a more forward authentication process [89]. Durey *et al.* [90] scrutinized 1,485 websites, with the objective of identifying security-related usages of browser fingerprinting. Their results show that 446 domains use browser fingerprinting for either authentication or for securing their payment processes. Of course, browser fingerprinting for security is not without its own pitfalls. Vastel *et al.* [12] show that a motivated attacker can generally defeat bot detection mechanisms that rely on fingerprinting, and Lin *et al.* [91] describe how attackers, through for example phishing sites, can collect fingerprints in order to defeat security features. Some work has been done on the use of dynamic tests to resist such replay attacks [92], but this approach does not appear to have taken off at the moment.

To date, browser fingerprinting has established itself as a complementary technique to cookie-based tracking, especially with the upcoming deprecation of third-party cookies.[11] However, the tracking potential of browser fingerprints is highly dependent on the presence of highly unique and stable attributes. As we present in chapter 3, we developed a technique to fingerprint GPUs through the WebGL API and show that it increases the effectiveness of browser fingerprint tracking.

[11] `https://developers.google.com/privacy-sandbox/blog/cookie-countdown-2023oct`

Table 2.1: Subset of attributes used for browser fingerprints

Attribute	**Source**	**Example**
User-agent	HTTP Header	Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0
Accept	HTTP Header	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Content Encoding	HTTP Header	gzip, deflate, br
Languages	HTTP Header	en-US,en;q=0.5
User-agent	JavaScript	Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/119.0
Canvas	JavaScript	Cwm fjordbank glyphs vext quiz, 😃 Cwm fjordbank glyphs vext quiz, 😃
Local storage	JavaScript	Yes
Screen resolution	JavaScript	1920x1080x24
Timezone	JavaScript	UTC+01:00
Hardware concurrency	JavaScript	12
WebGL	JavaScript	

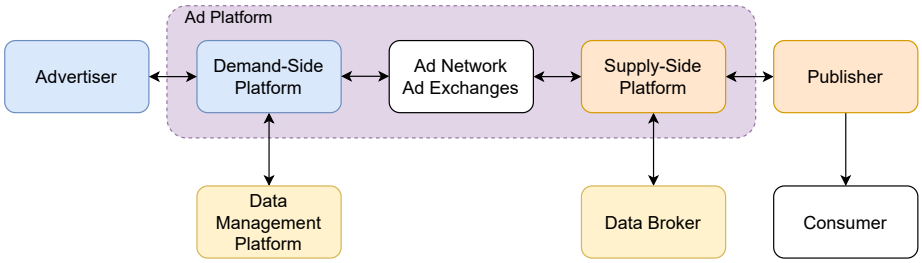


Figure 2.3: Actors of the Advertising Ecosystem

2.4.2 Advertising on the Internet

Actors of the online advertising ecosystem

The advertising ecosystem has historically been centered around physical ad agencies, where representatives looked to purchase ad space and attracted advertisers who were willing to provide payment to display their products in these spaces. Popular means of advertising was through newspapers or television. For newspapers, the process looked much like the current online advertising process: publishers (in this case, the newspaper), dedicated space that was sold to advertising agencies, which then sold this inventory to advertisers.

Online advertising, when simplified, relies much on this same principle and can be separated into four main interacting entities [93]:

- The **advertisers**, which generate the demand by their interest in promoting an item or a service. Advertisers look forward to displaying their ad on some part of a website and can either sign direct agreements with a website or buy inventory from ad platforms. More recently, advertisers have been able to participate directly in the advertising process through bidding strategies, such as *Real Time Bidding (RTB)*.
- The **ad platforms** represent a group of different entities that act as intermediaries between the demand and the supply. Their rise in popularity has been motivated by the complex state of the web, as it was becoming increasingly complicated for both publishers and advertisers to maximize their reach. Ad platforms have been introduced as a way to simplify the interaction between publishers and advertisers.
- The **publishers** are entities that provide space on their online application (i.e. a web page, a mobile application). Through their content, they draw the attention of users who then come into contact with the online advertising ecosystem through the ad slots present in the web application.
- Finally, **the users** see the ad and, if attracted by the publisher's content, can often click on the advertising and are brought to the product or service to potentially make a purchase.

Due to the increasing complexity of the online advertising ecosystem, ad platforms are now comprised of various entities that are oriented toward specific actors in the advertising industry and serve specific roles. We describe each of the major entities comprising the *ad platforms* in the following:

Ad Networks. Ad Networks are platforms that serve as intermediaries between publishers and advertisers. Such networks were the first brokers to appear in the online advertising ecosystem as a consequence of the increasing complexity of the Web. Today, ad networks mainly appeal to publishers that wish to sell their *remnant inventory* that could not be sold through more current platforms.

Ad Exchanges. Ad Exchanges are similar to Ad Networks, as they act as a broker between publishers and advertisers. However, Ad Exchanges sell their inventory through an auction system, which is considered more transparent to the publisher. The auction system also allows publishers to optimize their revenue, as multiple advertisers have the opportunity to place a higher value on ad-slots for a given user. The attractiveness of an Ad Exchange is mostly defined by its tracking ability: as it provides context about the users that generate the impression, the bids, and their values are influenced by the relevance of the user's profile that is shared by the Ad Exchange.

Demand-side platform (DSP). The DSP, as its name indicates, is oriented toward advertisers and acts as an aggregator of the various *Supply-side platforms (SSPs)* and Ad Exchanges by providing a single interface for advertisers to run their advertising campaigns [94]. DSPs are key to bidding processes such as *Real Time Bidding* and also provide tracking value by either providing their own user-collected data, or by purchasing user data on the market, and can therefore boost effectiveness for advertisers [95].

Supply-side platform (SSP). SSPs work similarly to DSPs but are oriented toward the supply, which corresponds to publishers. SSPs are designed to help publishers optimize their inventory management and enrich the user-tracking information that is provided to Ad Exchanges during auctions.

These entities act to ensure that either publishers or advertisers attempt to optimize their reach and revenue. However, with the advent of *programmatic auctions*, such platforms require a significant amount of user data in order to precisely target each ad to specific users based on their web activity and profile. As a means to acquire this data, big actors such as *Google*, *Meta* or *Amazon* make use of their web presence [96] and their platforms to track users and collect such information. Smaller actors can also choose to purchase this data. *Data brokers* have been introduced to provide standardized platforms to buy user data that is collected or bought from various data holders. Ad platforms can then buy such aggregated data by specifying their requirements or demographics, such as gender, age, interest, salary, or locality. Venkatadri *et al.* [97] investigated the coverage and accuracy of data brokers by leveraging Meta's advertising and found that a significant majority of Meta users (up to 90%) can be linked to some data broker information, showcasing the reach of their databases. Due to the enormous amount of data they handle, extensive literature exists on privacy issues linked to data brokers. Pinchot *et al.* [98] performed an explorative analysis of the potential impacts of data provided by data brokers and warns against potential misuse of the collected data. Yeh *et al.* [99] suggests in their study that the potential for misuse of the collected data should lead to stronger regulations: they state that regulations in the US should model current

European regulations, such as the GDPR (section 2.4.3), which offers strong privacy protections against misconduct exhibited by data brokers.

Advertising methodologies

The early days of online advertising were mostly characterized by direct interaction between publishers and advertisers. However, the exponential growth of the industry led to the emergence of various *programmatic* selling strategies that take advantage of the newly acquired tracking capabilities. To date, there exist four main strategies to sell and buy ad inventory. Publishers can choose to use one or multiple strategies, or rely on a *waterfall* process that prioritizes higher-yielding processes.

Direct deals. Originally, publishers and advertisers interacted directly and established an advertising contract in case both interests were corresponding. Advertisers establish direct deals with publishers depending on multiple variables such as the user traffic, the publisher's area of interest, and the expected *click-through rate (CTR)* (i.e. the amount of clicks on the displayed ad). However, despite remaining popular until the mid-2000s, the *direct deal* strategy started showing its limits as the Web grew and became more fragmented. Many publishers, including prestigious publishers, struggled to fill every ad slot through direct deals, prompting the need for an improved selling strategy [100], such as Real-Time Bidding.

Programmatic direct. Programmatic advertising emerged as a strategy to reduce human interaction and optimize both advertisers' and publishers' revenues. Programmatic advertising has been enabled by multiple factors, including increasingly growing computing capabilities, less expensive data storage, or the advent of mass online tracking [101]. It is characterized by the automation of the inventory sale process and is data-driven, enabling advertisers to precisely target their campaigns and track their performances. Programmatic Direct relies on the programmatic advertising principle and helps advertisers target publishers through dedicated platforms based on various criteria, such as the publisher's user base and its relevance to the advertiser's campaign. As advertisers are able to precisely target specific groups of users, this leads to a higher *Cost Per mile (CPM)* which benefits the publishers.

Real-time bidding (RTB). As the number of websites kept growing, many Ad Networks were faced with more supply than demand, leading to lower advertising revenue for publishers that could not assign all their impressions to interested advertisers. Advertisers attempted to overcome this issue by registering with multiple Ad Networks in an attempt to find lower-priced inventories. However, as a single advertiser subscribed to multiple Ad Networks, optimally managing the distribution of their ads became problematic. For publishers, subscribing to multiple Ad Networks meant a higher latency, while still not guaranteeing that their entire inventory would be sold. This led to the birth of Ad Exchanges during the late 2000s, which handled the increasingly complex interaction with multiple Ad Networks (section 2.4.2). Through Ad Exchanges, the RTB process was born and consisted of interrogating all potentially interested Ad Networks each time a user triggered an impression. RTB is based on an

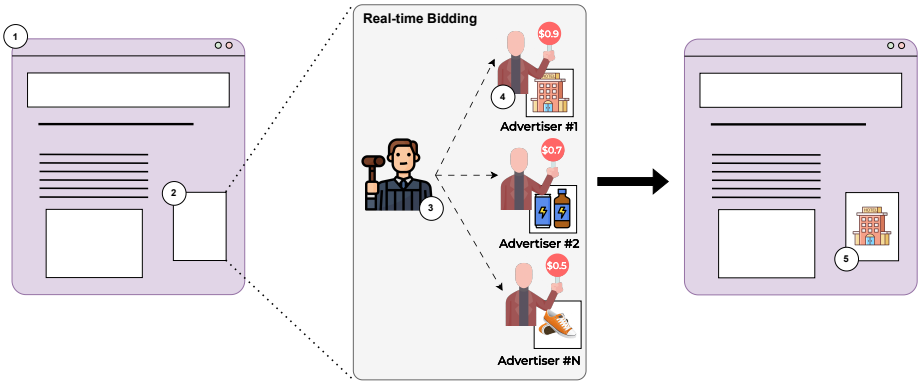


Figure 2.4: Overview of the Real-time Bidding process: (1) the user loads the website in their browser; (2) upon loading, the ad slot initiates the RTB process; (3) the auction begins by contacting the various advertisers and waiting for their bids; (4) the highest bidder wins the auction and forwards his ad; (5) the winning ad is displayed to the user.

auction process: as soon as a user accesses content from a publisher that is managed by the Ad Exchange, an auction is initiated by contacting all potentially interested DSPs. The transmitted information might contain contextual information along with user-related tracking data. The Ad Exchange then collects all bids from the contacted entities and picks the winner by ranking all bids and eliminating bids under a preset *floor price*. The Ad Exchange then proceeds to serve an impression of the winning ad to the publisher, which is then displayed to the user [102]. Muthukrishnan *et al.* [102] outline multiple problems with the way auctions are performed in Ad Exchanges. Notably, the authors discuss issues such as the truthfulness of auctions [103], call out optimization [104] or arbitrage bidding. To reduce the complexity of the entire process, advertisers are currently mainly relying on DSPs (section 2.4.2) to interact with Ad Exchanges, as they provide an aggregated interface with finer-grained impression reports and better customization abilities, allowing them to target specific groups of users [105]. On the other hand, publishers rely on SSPs (section 2.4.2) to handle and customize their impressions through Ad Exchanges, by providing them with the ability to set a specific price for a specific placement or a specific bidder. As a way of standardizing the RTB process, the *RTB Project* (formerly the *OpenRTB Consortium*) was created[12] and introduced the OpenRTB communication protocol, which proposes standardized and consistent ways to communicate auctions and receive bids. The process is depicted in Figure 2.4.

While RTB allows publishers to maximize their revenues, it suffers from multiple issues. One problem is the lack of transparency: at the end of the process, publishers are only aware of the winning bidders, which raises transparency issues. Another problem results from its *waterfall* nature: by contacting ad exchanges sequentially, there is no guarantee that the ad slot has been sold to the highest possible bidder. To solve this, *Header Bidding* (HB) [106] flattens the process and reaches out to all registered ad exchanges or *Supply-Side Platforms*

[12] `https://www.iab.com/guidelines/openrtb/`

(SSPs) at once. Header bidding also addresses the issue of transparency by giving entire control of the process to the publisher. As such, for each ad slot, the publisher defines which ad exchanges are contacted when an auction is started Header bidding comes in two main forms:

- *Client-side header bidding* happens entirely in the client's browser. The publisher defines each ad-slot and each publisher they wish to contact for each auction. When an auction starts, the client's browser initiates requests to the different publishers. Each request contains information about the ad slot, the floor price, and additional cookies about the user. Depending on their active campaigns and the provided information, the exchanges might decide to bid. In this case, they respond with an object containing the bid value and the ad's content. Once all bids are collected, the client identifies the auction winner and notifies the winning bid of the result. The ad is then displayed on the user's browser. While client-side header bidding provides many benefits to the publisher, it remains limited by the network performance of the client's device and browser.
- *Server-side header bidding* solves the issues of client-side header bidding by having the process happen entirely in a remote server rather than in the client's browser. The remote server remains however in complete control of the publisher. As such, both processes are similar but the amount of contacted exchanges is not limited by the client's hardware and software. However, as server-side header bidding does not happen in the client's browser, it doesn't include client-related cookies and therefore transmits a less precise profile of the user to advertisers.

Both forms of header bidding have gained increasing popularity in the last years: in 2019, Pachilakis *et al.* [107] have shown that over 14% of the Top *35,000* Alexa websites implemented client-side header bidding.

Private marketplace (PMP). Despite the introduction of the RTB process, many publishers with premium inventories felt that the RTB process did not yield sufficient value for their ad slots. Advertisers felt similarly, estimating that many of their ads were not displayed at premium spots and were therefore not as impactful to users. As a solution to these issues, the *Private marketplace (PMP)* process was proposed as a variation of the RTB process but based on an invite-only system for publishers to offer their premium inventories to a reduced amount of buyers. For publishers taking advantage of such a system, the PMP process takes precedence over the RTB process, ensuring that all premium inventory is sold at optimal value.

2.4.3 Regulations & Defenses

In the digital age, user privacy has gotten significant exposure thanks to growing concerns against data hoarding by the Web community. As online services have become omnipresent, the collection and exploitation of personal information have raised significant concerns about safeguarding fundamental rights and freedoms. In response to this, regulations have emerged all over the world, as an attempt to balance legitimate business practices and upholding data sanctity. Among these, the European Union's General Data Protection Regulation

(GDPR) has ushered in a paradigm shift reverberating globally. This section explores the intricate landscape of existing regulations and their impact, before delving into the different possible defenses against online tracking.

Existing regulations

In light of the privacy and security threats of the Internet, various actors have introduced regulations and tools to limit the impact of tracking and advertising. The most renowned regulation that durably changed the face of the Web is the **General Data Protection Regulation (GDPR)** [108]. The GDPR is a European legislation that requires every aspect of interaction with personal data to be carefully planned: the regulation attempts to answer all questions related to handling, processing, and storing of personal data by introducing legislation for all each case. While the regulation was adopted in 2016, the European regulator granted businesses until May 2018 to fully comply with its principles. One of the major advantages of the GDPR is that it is not purely limited to businesses present on European soil but covers any institution that interacts with the personal data of European citizens, who make up a significant portion of Internet users. Incidentally, most businesses, regardless of their location, have been forced to comply with the GDPR. The main goal of the GDPR is to give users control over their data, while at the same time unifying the various regulations that were present in different European countries prior to the law. The regulation is composed of six key principles that include lawful processing, purpose limitation, data minimization, data accuracy, storage limitation, and data security. One essential concept is that of consent. This implies that activities to profile users require informed consent from the user before they can collect, process, or sell the data.

The GDPR has been successful in introducing a change in how businesses process personal data, due to improved coverage and the existence of significant fines in case of non-compliance. Fines under the GDPR can reach 4% of the total worldwide revenue of a company. These changes were outlined in various studies: in 2019, Degeling *et al.* [109] analyzed the different changes that the privacy regulation introduced in the 500 most popular websites of each European country and found that 15.7% of websites added new privacy policies as a result. Their study also outlined that 16% more websites introduced a cookie banner once the regulation was put into effect. Linden *et al.* [110] pursued a similar study but also included non-European websites: their results show that contrary to previous privacy regulations, the increased length of privacy policies due to the GDPR has not resulted in decreased readability but instead allowed a better coverage of relevant topics, along with better compliance to its principles.

The post-GDPR world is filled with notices asking for explicit consent to store various types of cookies other than *strictly necessary cookies*, each having different purposes.[13] Sanchez *et al.* [111] analyzed how the GDPR impacted cookies and cookie notices: at the time of their study, their findings stated that 92% of the visited websites started tracking users even before serving any sort of cookie notice, and only 4% of them truly entirely respect the users' choice when they

[13] `https://gdpr.eu/cookies/`

opt out of cookies. These results are further confirmed by more recent studies [112, 113, 114]. While these studies show that many websites tend to take advantage of the GDPR's loopholes, the regulation has durably changed the face of the web and paved the way to better protection of personal data both in Europe and the World. It led to multiple countries developing their own privacy regulations: in California (US), the **California Consumer Privacy Act (CCPA)** was enacted around the same time as the GDPR but came into effect only in 2020. Inspired by the GDPR, the law introduced tight regulations [115], including the right to know what information businesses collect and share about users, to opt-out, and to ultimately delete this data. The exertion of these rights is also subject to the right of non-discrimination, as stated in the regulation. In Brazil, the **General Data Protection Law (LGPD)** started in August 2020, with close key principles to the GDPR's.

Nevertheless, the GDPR is not the first regulation to frame the use of personal data of European citizens: in 2002, the European Parliament enacted the **ePrivacy Directive**, which ensured that all communications over public networks maintain respect fundamental rights. Its principles were broader than the current GDPR. The directive states the right for individuals to determine for themselves what is communicated to others, might they be businesses or individuals. The ePrivacy Directive was amended in 2009, in light of the fast evolution of the Internet ecosystem: since then, it has been dubbed *the cookie law* because, similarly to the GDPR, it imposed websites to inform users before storing any cookies, except for strictly necessary cookies. This led to the birth of cookie notices, prompting users to consent. However, the scope of the directive was more restrained than the GDPR's, leading to a less visible impact. To further adapt to the new challenges posed by the Internet, the **ePrivacy Regulation** is currently being drafted as an extension to the GDPR and will take precedence over it [116]. The ePrivacy Regulation has been in talks since 2017 and is expected to cover all electronic communication, including those that involve no personal data. It is expected to lay down a series of rules regarding the protection of fundamental rights and freedom of both natural and legal persons. The reasoning behind the broader range of application of the upcoming regulation lays in the fact that all potential communication might leak personal information about the users: for instance, metadata might reveal sensitive information such as the dialed phone numbers, visited websites or location data. It is also expected to cover cookies and tracking in general by simplifying existing rules. Similarly to the GDPR, the ePrivacy Regulation might lead to fines of up to 4% of a business's global annual turnover and is expected to help enforce the fines more effectively.

As the digital landscape continues to evolve, new privacy-oriented regulations attempt to find a balance between protecting user privacy and enabling legitimate business practices, while also highlighting that current practices need clear and enforceable guidelines to govern the constantly evolving data collection practices.

Defenses Against Online Tracking.

The various threats faced by Internet users with regards to their privacy led to increased awareness [117], which in turn encouraged various actors to take

action. Browsers, initially focused on injecting new features without much regard for the privacy implications of such features, started putting privacy in the heart of their product. Currently, all major browsers engines come with many privacy measures by default: the most popular is *private browsing*, first introduced by Apple's Webkit engine, is a standardized tool that initiates temporary browsing sessions that are erased once the browsers' private windows have been closed [118]. Additionally, each of the major browsers provide options to entirely disable third-party cookies through privacy settings that can be toggled. The *Referer-Policy* HTTP header is also currently supported by all browsers and has been designed to mitigate the capacity of third-parties to build browsing histories of visiting users through their *Referer* header. *Do Not Track (DNT)* is another HTTP header designed to restrict tracking on the Web. However, this initiative did not yield the expected results. Furthermore, browser engines have introduced other privacy-related features:

- Safari, Apple's WebKit-based browser, provided by default on Apple products, originally introduced their *Intelligent Tracking Prevention (ITP)* program,[14] which is mainly aimed at protecting users from tracking by preventing cross-site information sharing. ITP works by establishing an on-device list of prevalent domains that are contacted by visited websites: it includes a classifier that helps decide whether the contacted domain performs cross-site tracking and a blocking decision is then reached [119]. ITP has evolved and been improved since it was first introduced in 2017: in 2020, Apple announced that it would be integrating its CNAME-cloacking [11] defense in its most recent version of ITP.[15]

- Firefox, driven by the Gecko browser engine, initially introduced its *Enhanced Tracking Prevention (ETP)* feature,[16] which blocs third-party trackers and cookies based the Disconnect filter list.[17] ETP comes with three different levels of protection: `Standard`, `Strict` and `Custom`. In the `Strict` mode, an increased number of in-page trackers are blocking following the same principle as the `Standard` Mode. More recently, as an improvement to ETP, Firefox started rolling out its *Total Cookie Protection*, which is presented as their strongest privacy protection to date. It works by isolating each cookie created by a website, including third-party cookies, to the website they were created on. This feature is currently enabled by default on recent versions of Firefox and presents the advantage of not relying on filter-lists.

- Google Chrome, based on the Chromium engine, started work on its *Privacy Sandbox* initiative in 2019. The Privacy Sandbox is a set of experimental tools that are designed to ultimately replace third-party cookies, allowing advertisers to continue to provide targeted advertisements while limiting the impact on user privacy at the same time. Examples of the proposed APIs include the Topics API, which was included as a replacement for the FLoC API following significant criticisms raised by online

[14] `https://webkit.org/tracking-prevention/`

[15] `https://webkit.org/blog/11338/cname-cloaking-and-bounce-tracking-defense/`

[16] `https://blog.mozilla.org/en/mozilla/firefox-rolls-out-total-cookie-protection-by-default-to-all-users-worldwide/`

[17] `https://disconnect.me/trackerprotection`

communities and experts. The Topics API suggests associating users to a weekly list of topics of interest (pooled from a predefined and fixed list) that the advertisers can later use to target their advertisements. The Chromium engine also includes other tools that are designed to reduce the privacy risks of Web browsing: since Chrome 85, the browser incorporates *Cache Partitioning*, which prevents cache-base timing attacks that potentially reveal users' browsing history. To do this, Chrome stopped using the request's URL as a key to access the cached value and now uses a combination of the URL and a network isolation key. The cache access strategy has also been changed from a per-resource only strategy to a combination of both per-resource and per-website strategies. This same version of Chrome reinforced compliance with the *Referer-Policy* HTTP header by setting its default value to *strict origin when cross origin*,[18] which specifies that the current full URL is sent in the *Referer* header only when the destination URL has same origin (same scheme, hostname and port) and uses the same protocol.

Moreover, there exists various privacy-focused browsers, such as the *Brave* browser,[19] which is based on a fork of the Chromium engine and integrates an ad-blocker (*Brave Shields*) and various privacy preserving tools present in the Chromium engine that are enabled by default [120]. In [121], Leith studied the telemetry transmitted by each browser during use: their findings show that the Brave browser is the only one that shows no evidence of sending identifiable telemetry back to its servers. The Tor project[20] was initiated by US Naval Research Laboratories in 1995 [122]. It resulted in the Tor Browser, a privacy-focused browser based on a modified version of the Gecko engine that is altered for stricter privacy measures. The Tor browser also leverages the Tor network to ensure the anonymity of its users. The Tor network consists of a global network of relays that ensures anonymity of their Internet traffic. Upon initialization, the Tor browser initiates a *virtual circuit* consisting of a default of three successive relays randomly picked by the browser. Their routing information is downloaded to the user's node and encryption keys are exchanged using the Diffie-Hellman key exchange protocol. Every packet that is transmitted is encrypted multiple times using the encryption keys of each of the relay nodes. Packets enter the network through a *guard relay* and are relayed through middle relays until reaching the *exit relay*. The packets are successively decrypted by each node before reaching their *exit relay*, which obtains access to the final destination's information [123, 124]. In the case of HTTPs communications, the packet's content remains encrypted until its final destination, the website the user is attempting to access. The *virtual circuit* ensures that the user's IP address is not known by the exit relay nor the website being accessed and that collusion between the guard relay and exit relay is difficult. To ensure anonymity, the Tor browser picks a new *virtual circuit* every 10 minutes. The Tor browser itself also provides a series of privacy-preserving features. For instance, third-party cookies are strictly isolated through *double keying*: through this process, cookies are isolated both on the first and third-parties' origins, preventing third-party cookies from being accessed from different websites, while

[18] `https://developer.chrome.com/blog/referrer-policy-new-chrome-default/`
[19] `https://brave.com`
[20] `https://www.torproject.org/`

at the same time reducing breakage, since cookies are still stored and retrieved when requested. Browser fingerprinting is also mitigated through uniformity measures. Some measures include disabling plugins, canvas mitigation strategies, a fixed list of fonts, and click-to-play WebGL canvases, among others [125].

Browser extensions are third-party software developed using standard web technologies, such as HTML, CSS, and Javascript [126]. They introduce new features to browsers and increase usability. Community-based efforts to improve privacy on the web and reduce the nuisances caused by online advertising have led to significant use of browser extensions to introduce privacy-preserving features. One such example is ad-blockers. Ad-blockers are currently among the most popular browser extensions on all major browsers: on the Chrome Web Store, *Adblock Plus*, one of the oldest ad-blockers available, accounts for over $47,000,000$ users. On Firefox, over $4,100,100$ browsers are actively using AdBlock Plus. Various privacy-preserving extensions exist: *uBlock Origin* is an extension for content filtering that is mainly used as an ad-blocker. *Privacy Badger* is another tracker-blocking extension that uses automated learning techniques to block invisible trackers and remove advertising. *Decentraleyes* is an extension that distribute website's resources from the local cache instead of collecting them from a centralized content delivery service, which would aid in tracking the user.

Most ad-blocking extensions use community-maintained *filter-lists* to identify and block tracking and ad-related network requests. Filter lists are files consisting of rules targeted at blocking URLs through a syntax closely related to regular expressions. The rules can include exceptions, designed to reduce breakage due to blocking. Traditional filter-lists include *Easylist*, *Easyprivacy*, or the *Disconnect*. *uBlock Origin*, while supporting traditional filter list syntax, expands on those and introduces scriptlet support in their filter lists that can be used to disable specific Javascript code on a website. An example of scriptlet can be found in the blocking strategy employed against the *Youtube platform*. Filter-lists may also be used to introduce DOM-altering rules by collapsing, removing, or hiding parts of the HTML content in the page to reduce the nuisance introduced by ads.[21]

Prior to the widespread use of ad-blockers, *hosts* files were used as a blocking mechanism to prevent tracking requests. The structure of a hosts file consists of two columns, the first column mentions the hostname and the second specifies the IP address that the hostname is supposed to point to. When they are used in order to block network requests, hostnames are mapped to a local IP address, most commonly *127.0.0.1*, causing the request to fail with very low latency. The disadvantage of hosts files lies in the fact that they cannot block requests at finer granularities than the subdomain, as opposed to rules in filter lists. Despite this disadvantage, they remain a tool that is commonly used against tracking, most notable for devices that do not support browser-based ad-blockers, such as Smart TVs. Two popular examples of such lists are the *Peter Lowe's list*[22] or the *MVPS hosts* file.[23]

[21] `https://github.com/gorhill/uBlock/wiki/Static-filter-syntax`
[22] `http://pgl.yoyo.org/adservers/`
[23] `http://winhelp2002.mvps.org/hosts.htm`

DNS sinkholes, such as *Pi-hole*[24] or the *AdGuard DNS*,[25] are also commonly used for ad-blocking in unconventional devices, such as smart TVs. They act as blockers at the DNS-level by preventing DNS requests from resolving when used as the device's DNS server. These tools can be self-hosted for private use and source their advertising and tracking domains' database from popular hosts files.

Ad-blockers that do not perpetuate the filter list usage have also been made available. One such example is *Privacy Badger*, which uses automated learning to detect tracking behaviors that are not listed in common filter lists. These ad-blockers use heuristics and machine learning to identify domains that are frequently contacted by the browser and then block them.

Moreover, ad-blockers are not the only available extensions whose purpose is to improve privacy and reduce annoyances on the web. For example, cookie editors allow users to alter and set up automatic deletion of their cookies after a browsing session. Furthermore, extensions such as *Consent-o-matic*[26] use community-maintained lists to explicitly refuse specific categories of cookies. *I Don't Care About Cookies* is an extension designed to automatically remove cookie banners without explicitly refusing them. It uses a series of selectors to generate click sequences that remove cookie banners on a selection of websites. As this extension might sometimes accept cookies, it should be used in conjunction with a cookie eraser.

Other recent developments of ad-blocking have explored the potential of perceptual ad-blockers, which leverage visual features to detect advertisements on webpages [127], but such methods are prone to adversarial evasion techniques and are generally perceived as less robust than the traditional filter-lists based blocking techniques [128].

Finally, *Virtual Private Networks (VPNs)* and *proxies* can also be used as privacy-preserving tools by hiding the users' IP addresses and even blocking requests at the network level. VPNs from trusted vendors [129] offer an additional layer of privacy by routing the user's internet traffic through an encrypted tunnel, protecting it from potential surveillance and eavesdropping attempts.

Despite the efficiency of these tools, advertisers have been shown to make use of circumvention techniques to bypass the various protections. One such example is known under the name of *CNAME Cloaking* [11, 130], in which the CNAME record is altered to hide the true destination of tracking requests, effectively masking the identity of third-party services. In most cases, the third-party service is cloaked under an alias or subdomain of the visited website, evading detection by ad-blocking tools.

In their study, Chen *et al.* [131] exploited the Blink and V8 Engine in Chrome to construct signatures of JavaScript functions, in an attempt to detect evasion techniques in the tracking ecosystem. They find that most evasion techniques consist of moving the tracking URL to another URL that has not been flagged yet by ad-blockers. Inlining code is also a common evasion technique consisting

[24] `https://pi-hole.net/`
[25] `https://adguard-dns.io/`
[26] `https://consentomatic.au.dk/`

of including tracking scripts directly into the webpage's HTML code, removing the need to perform any HTTP requests that can be blocked. However, some ad-blockers, such as *uBlock Origin* include scriplet blocking, which can mitigate this technique. Finally, trackers might bundle malicious code with benign code, leading to page breakage if ad-blockers prevent the resource from loading.

2.5 Carbon Impact of Online Advertising

It is estimated that the Internet uses over 10% of global electricity and the trend is rising [132, 133]. However, measuring the carbon impact of Internet services remains a difficult task. The reasons behind this difficulty include a lack of access to relevant and up-to-date information, along with a high diversity of materials and infrastructure used worldwide. The Internet runs mainly on black boxes: services are requested and distributed to the requester. Requests are routed through non-deterministic paths in the network, depending on various parameters such as traffic load and availability. The requested content can be cached or computed on the fly, without much information on the underlying software. Finally, it is impossible to remotely identify every device, material, computer router or node being used in every step of an HTTP request without being granted access by those responsible for this infrastructure.

Nevertheless, a few studies have attempted to estimate the carbon impact and electricity consumption of the Internet. In most studies, the Internet is separated into three main components: the client devices, the network infrastructure, and the server (data centers) infrastructure. Baliga *et al.* [134] used data from major equipment vendors to model the carbon impact of the network infrastructure. Using their model, which takes into consideration switching and transmission equipment, they estimated in 2009 the energy consumption of the Internet in Australia to be about 75 KWh per ISP subscriber yearly, translating to 81 $kgCO_2eq$ every year . In 2010, Malmodin *et al.* [135] performed a lifecycle assessment (LCA) of the Information and Communication Technologies (ICT) industry in Sweden using data obtained from Swedish operator TeliaSonera. Lifecycle assessments provide a global view of the environmental impact of devices from cradle to destruction. Furthermore, the scope of the study is quite large, encompassing end users' devices along with network infrastructure and data centers. Their study finds that 1.5 $MtCO2_e$ is emitted for the ICT industry in Sweden , which represents approximately 160 $kgCO2_e$ per Swedish citizen . They estimate that user devices are the biggest source of environmental impact, with over 50% of the total carbon impact being attributed to them. In 2013, Corcoran & Andrae [136] present the electricity consumption trend for consumer ICT and estimate that new devices are expected to drive the consumption of ICT down from 7.4% of the total global electricity consumption in 2012, to 6.9% by 2017. In their worst-case scenario. However, this number could rise to 12% due to the expansion of networks and data centers. In 2018, Malmodin *et al.* [137] extended their analysis to estimate the global carbon impact of the ICT industry, along with the entertainment and media industries, from 2010 to 2015. They estimate that the worldwide ICT industry consumed 805 TWh in 2015, which equals to 730 $MtCO_2e$, representing 1.4% of worldwide carbon emissions. However, Malodin *et al.* expect this number to decrease, as

both hardware and software are gaining in efficiency. Today, it is estimated that the Internet uses over 10% of global electricity, and while some work forecast a decrease of this share, others expect this percentage to keep rising in the future years [132, 133].

While the previous studies provide a glimpse into the macro-environmental impact of the Internet, the literature is lacking when it comes to the carbon footprint of the online advertising industry. To date, two studies present life cycle assessments of the industry. In 2008, Taylor *et al.* [13] were the first to attempt to measure the carbon footprint of online advertising. Their work provides a range for the carbon impact of the industry, between 256 $kgCO2_e$ and 676 $kgCO_2e$ per million impressions. In 2018, Pärssinen *et al.* [138] perform a more extended lifecycle assessment providing insights into both the global carbon impact of the ICT and the carbon footprint of the online advertising industry. They find that online advertising is responsible for electrical consumption ranging between 791 and 1334 TWh. Using a global averaged emission factor of 0.5656 $kgCO_2e/kWh$, they find that the industry is responsible 60 $MtCO_2e$ of emissions. Pärssinen *et al.* also estimated the share of "fraudulent advertising" to 13.87 $MtCO_2e$ in 2016. While both of these studies provide significant insight into the share of online advertising in the total carbon consumption of the ICT, these results are difficult to use or interpret at finer granularities, for example, at the scale of browser requests.

To solve this issue, and motivated by the need to provide a comprehensive framework to raise awareness on the carbon footprint of the Internet, the *1byte Model* was presented by The Shift Project.[27] Their model basically breaks down the whole system into three major parts: the client's device, the network, and the data center. Each of these components is assigned a constant value that represents their average electrical consumption per byte. However, as can be expected, these models lack precision as they remove a lot of variables from the equation. For example, user devices are diverse and their consumption varies greatly. Data-center consumption might not only be based on the amount of information they handle but also on the time they spend handling this information. Finally, network usage includes many variables, with the number of routers a packet is sent through being just one of them. Due to the difficulty in handling these variables, very few studies have provided a fine-grained look at the carbon impact of the online advertising industry. In 2020, CarbonTag [139] provides one of the first glimpses into the carbon footprint of ads at the network request granularity: based on various ads measurements, the authors design a machine-learning model that estimates the carbon footprint of an ad-related request. Through their measurements, they estimate that a single ad emits between 5e-7 and 1e-5 gCO_2e. While this study provides interesting and novel results, its scope is limited and focuses only on the client's device. In their paper, the authors acknowledge the need for a framework that covers the end-to-end ad process.

In Chapter 5, we leveraged client, network, and data-center measurements to introduce AdCarbon, a tool to estimate the end-to-end carbon cost of individual network requests, which we use to estimate the carbon footprint of online advertising.

[27] `https://theshiftproject.org/en/home/`

Chapter 3

GPU Fingerprinting

Privacy is dignity. It is a human right. In the domain of web browsing, the right to privacy should prevent websites from tracking user browsing activity without consent. This is the case in particular for cross-site tracking, in which website owners collude to build browsing profiles spanning multiple websites over extended periods of time. Unfortunately for users, the right to privacy conflicts with business interests. Website owners are highly interested in tracking users for the purpose of showing them ads they are more likely to click on, or to recommend products they are more likely to purchase.

As stated in Chapter 2, a significant difficulty of fingerprint-based tracking is that browser fingerprints evolve. As shown by Vastel *et al.* [140], fingerprints change frequently, sometimes multiple times per day, due to software updates and configuration changes. To track a user, an adversary must link fingerprint evolutions into a single coherent chain. This process is made difficult by the existence of devices with identical hardware and software configurations. It is difficult for an adversary to correctly link a fingerprint if there is a set of identical devices to which it might belong. This limits the adversary's tracking duration. In Vastel *et al.* 's evaluation over a dataset of nearly $100,000$ fingerprints collected from $1,905$ distinct browser instances, with a wide variety of fingerprinting attributes, their state-of-the-art machine learning technique was able to deliver a median tracking time of less than two months.

In this work, we bring a new insight to the challenge of browser fingerprinting identical computers, by observing that even nominally identical hardware devices have slight differences induced by their manufacturing process. These manufacturing variations are shown to enable the extraction of unique and robust fingerprints from a variety of devices, both large and small, in other settings [141, 142]. If an adversary were able to extract such a hardware fingerprint from the user's device, it would significantly extend the adversary's ability to track them. Extracting a hardware fingerprint from a browser, however, is far from trivial—since the attacker has little control. In particular, the attacker can only interact with the system through unprivileged JavaScript code and WebGL graphics primitives—the attacker has no control over the runtime environment of the system, including background processes and simultaneous user activity—and the attacker has very limited exposure to the system, making classical machine learning pipelines that rely on long training phases all but useless. Thus, in this chapter, I present our work on DrawnApart and show that browser fingerprinting can work on devices with similar hardware and software configuration.

DrawnApart measures small differences among the *Execution Units* (EUs) that make up a modern *Graphics Processing Unit* (GPU). By fingerprinting the

GPU stack, DrawnApart can tell apart devices with nominally identical configurations, both in the lab and in the wild. In a nutshell, to create a fingerprint, DrawnApart generates a sequence of rendering tasks, each targeting different EUs. It times each rendering task, creating a fingerprint trace. This trace is transformed by a deep learning network into an *embedding vector* that describes it succinctly and points the adversary towards the specific device that generated it.

We evaluate DrawnApart in two main scenarios. First, to validate the method's ability to distinguish nominally identical configurations, we perform a series of controlled experiments under lab conditions. We experiment with multiple sets of identical devices from vendors including Intel, Apple, Nvidia and Samsung, and demonstrate that DrawnApart consistently improves identification of these nominally identical devices, achieving high identification accuracy in multiple hardware configurations, even though state-of-the-art browser-based fingerprinting methods cannot tell them apart. Second, to show that DrawnApart affects user privacy, we integrate the technique into Vastel *et al.* 's state-of-the-art fingerprinting algorithm from IEEE S&P 2018 [140], which uses machine learning to link browser fingerprint evolutions. We show that the median tracking duration is improved by up to 66.66% once we add the DrawnApart fingerprint.

This chapter is organized as follows: in Section 3.2, we design and implement DrawnApart, a GPU fingerprinting technique based on the relative speed of EUs, that observes minute differences between GPUs. Then, in Section 3.5, we integrate DrawnApart into Vastel *et al.* 's fingerprinting algorithm and show, through a large-scale crowd-sourced experiment with over 2,500 unique devices and almost 371,000 fingerprints, that DrawnApart delivers considerable gains to the tracking accuracy of this state-of-the-art approach. Finally, we conclude in Section 3.6.2, by suggesting possible countermeasures against our fingerprinting technique, and discuss their advantages and drawbacks.

3.1 Background

3.1.1 Browser Fingerprinting

Mowery *et al.* [143] discuss fingerprinting on the Web. As they state, fingerprinting can be applied constructively or destructively. An example of constructive use of fingerprints would be to identify fraudulent users trying to log in while masquerading as legitimate users. Browser fingerprinting can be used to detect bots [144, 145, 146], or support authentication, where the fingerprint is used in addition to a traditional authentication mechanism [147, 148]. A destructive use might involve tracking users without consent [149, 150]. In this scenario, fingerprinting is used to augment or replace cookies—*e.g.* to track across multiple domains, or when users disable or delete cookies. Our technique can be applied to either scenario.

Many fingerprinting techniques exist in the wild [151, 152, 86, 153]. They rely heavily on differences in devices' hardware and software characteristics found in HTTP header fields and JavaScript attributes. The key challenge is to identify features and attributes that further discriminate devices and allow for their

unique identification, and to overcome the tendency of these features to evolve over time because of changes to the user's software, configuration, or environment.

3.1.2 GPU Programming

The *Graphics Processing Unit* (GPU) is specialized hardware for rendering graphics. GPUs have highly parallel architectures that are composed of multiple *Execution Units* (EUs), or *shader cores*, which can independently perform arithmetic and logic operations. Most consumer desktop and mobile processors from the past decade have on-chip GPUs with multiple EUs. For example, the UHD Graphics 630 GPU—integrated into Intel Core i5-8500 CPUs—includes 24 EUs, while the Mali-G72 GPU—integrated into the Samsung Exynos 9810 chipset used in Galaxy S9, S9+, Note9, and Note10 Lite devices—includes 18 EUs.

Web Graphics Library (WebGL) is a cross-platform API for rendering 3D graphics in the browser [154]. WebGL is implemented in major browsers including Safari, Chrome, Edge, and Firefox. Derived from native OpenGL ES 2.0, a library designed for developing graphic applications in C++, WebGL implements a JavaScript API for rendering graphics in an HTML5 canvas element. WebGL takes a representation of 3D objects as a list of *vertices* in space and information on how to render them, and translates them into a two-dimensional raster image that can be displayed on screen. WebGL abstracts this process as a pipeline. Two pipeline steps which are of interest to this work are the *vertex shader*, which places the vertices in the two-dimensional canvas, and the *fragment shader*, which determines the color and other properties of each fragment. The vertex and fragment shaders can run user-supplied programs, written in a C-derived programming language named *GL Shading Language* (GLSL).

3.2 GPU Fingerprinting

3.2.1 Motivation

Similar to past work [152, 155], we aim to uniquely identify devices. However, unlike previous work, which rely on the diversity of hardware and software configurations, we focus on distinguishing identical devices. As we show experimentally, this additional distinguishing power can considerably enhance the tracking capabilities of existing fingerprinting methods. To do so, we incorporate techniques similar to the arbiter-based *Physically Unclonable Function* (PUF) concept of Lee *et al.* [156]. In an arbiter PUF, the statistical delay variations of wires and transistors across multiple instances of the same integrated circuit design are used to uniquely identify individual instances of the integrated circuit. In our case, we harness the statistical speed variations of individual EUs in the GPU to uniquely identify a complete system.

3.2.2 Design

With unfettered access to the GPU, an adversary could measure the speed of each EU and use those measurements as a fingerprint. However, websites

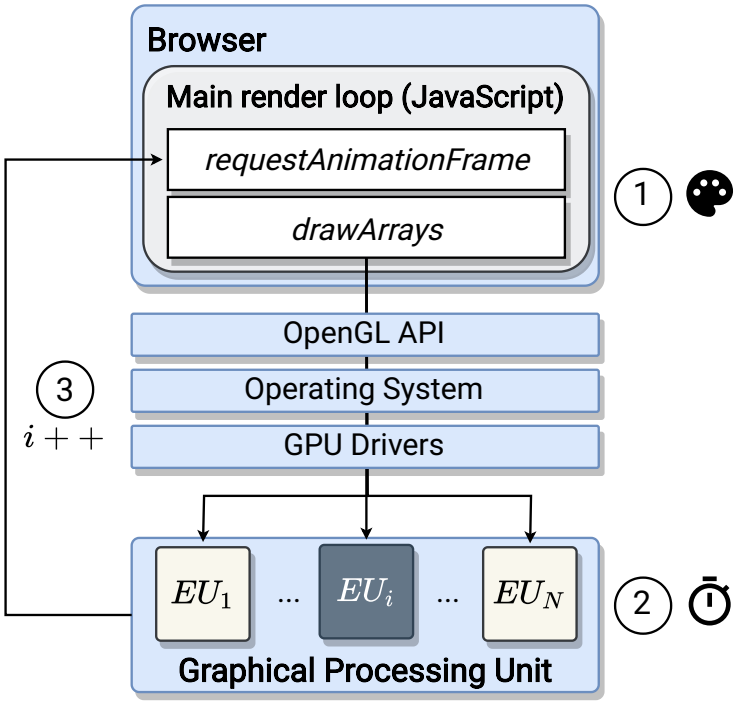


Figure 3.1: Overview of our GPU fingerprinting technique: (1) points are rendered in parallel using several EUs; (2) the EU drawing point i executes a stall function (dark), while other EUs return a hard-coded value (light); (3) the execution time of each iteration is bounded by the slowest EU.

only have limited access to the GPU through the JavaScript and WebGL APIs. WebGL provides a high-level abstraction that makes it a challenge to target specific EUs and to time computations accurately.

We overcome this challenge by using short GLSL programs executed by the GPU as part of the vertex shader (cf. section 3.1.2). We rely on the mostly predictable job allocation in the WebGL software stack to target specific EUs. We observe that, when allocating a parallel set of vertex shader tasks, the WebGL stack tends to assign the tasks to different EUs in a non-randomized fashion. This allows us to issue multiple commands that target the same EUs. Finally, instead of measuring specific tasks, we ensure that the execution time of the targeted EU dominates the execution time of the whole pipeline. We do so by assigning the non-targeted EUs a vertex shading program that is quick to complete, while assigning the targeted EUs tasks whose execution time is highly sensitive to the differences among individual EUs. As shown in fig. 3.1, our fingerprint is created by executing a sequence of drawing operations. We measure the time to draw a sequence of points with carefully chosen shader programs. The technique consists of three main steps:

Render. We instruct the WebGL API to draw a number of points in parallel. Points are the simplest object that WebGL can draw, and each consists of only a single vertex. Using points minimizes the noise from the pipeline and its

interference with our technique. The position of each point is determined by an attacker-controlled vertex shader.

Stall. For most points, the attacker-controlled vertex shader returns a hard-coded value. For a specific subset of the points the shader applies a function, which we call a *stall function*, to compute the point's position. The manner in which the entire graphics stack distributes the points to be drawn to the EUs allows us to influence which EU is chosen to run the stall function. It takes much longer to compute the position with the stall function than the hard-coded value. As a result, the time needed to render the entire set of points corresponds to the time taken by the EUs running the stall function.

Trace Generation. We execute the drawing command several times, each time selecting a different vertex to stall. For each execution, we store the time taken. The fingerprint output by our technique is therefore a vector, named a *trace*, which contains the sequence of timing measurements.

We note that prior browser fingerprinting techniques extract **deterministic** fingerprints, which remain identical as long as the device's software and configuration have not changed. Our technique, in contrast, is based on timing measurements and, as such, is **non-deterministic**—multiple measurements made on the same device will return different values due to the effects of measurement noise, quantization, and the impact of other tasks running at the same time.

3.2.3 Implementation

We now describe the implementation of each design step.

Render. The WebGL API exposes the `drawArrays()` function, which allows dispatching multiple drawing operations in parallel to the GPU. We invoke `drawArrays()` several times, each time rendering multiple points in parallel. Listing 3.1 describes our main render loop. We execute the rendering process by calling `drawArrays` (line 5). For each iteration, we save the time to execute `drawArrays` into the `trace` array. We evaluated several ways of measuring the rendering time, as explained further in section 3.4.1. Briefly put, the **onscreen** measurement method executes a relatively small number of computationally intensive operations, while the **offscreen** and **GPU** measurement methods execute a larger number of less computationally intensive operations. The full source code for these settings can be found in our artifact repository.[1] After `point_count` iterations, the code sends the `trace` array to our back-end server (line 15), and terminates the loop.

Stall. In the current implementation of WebGL, a single call to `drawArrays()` generates multiple drawing operations in the underlying graphics API, which appear to assign vertices to EUs in a deterministic order during vertex processing. The operations are differentiated by a global variable, named `gl_VertexID`. This special variable is an integer index for the current vertex, intrinsically generated by the hardware in all of the graphics APIs used to implement WebGL as it executes `gl.drawArrays`. We created a vertex shader in GLSL that examines the `gl_VertexID` identifier, and executes a computationally intensive *stall function*

[1] `https://github.com/drawnapart/drawnapart`

```
function render_loop() {
  if (point_index < point_count) {
      // Stall the current point
      gl.uniform1i(shader_stalled_point_id, point_index);
      gl.drawArrays(gl.POINTS,0,point_count);
      // Save the rendering time
      var dt = performance.now() - prev_time;
      prev_time = performance.now();
      trace.push(dt);
      // Prepare to stall the next point
      point_index++;
      requestAnimationFrame(render_loop);
  } else {
    // Finish and send the trace to the server
    send_trace();
  }
}
```

Listing 3.1: Main Render loop, onscreen setting (JavaScript).

only if it matches an input variable named `shader_stalled_point_id` provided by the JavaScript code running on the CPU. Listing 3.2 describes the vertex shader code.

```
uniform int shader_stalled_point_id;
void main(void) {
  // Stall on this vertex?
  if(shader_stalled_point_id == gl_VertexID) {
    gl_Position = vec4(stall_func(),0, 1,1);
  } else {
    gl_Position = vec4(0, 0, 1 ,1);
  }
  gl_PointSize = 1.0;
}
```

Listing 3.2: Vertex shader with stall function, onscreen setting (GLSL).

In the onscreen setting, the vertex shader checks if `shader_stalled_point_id` equals `gl_VertexID`. In the offscreen and GPU settings, the vertex shader treats `shader_stalled_point_id` as a bit mask and checks if bit `1 << gl_VertexID` is set. In both cases, if the point is selected the vertex shader program executes the stall function (line 5). Otherwise, the shader exits quickly.

Trace Generation. By executing this parallel drawing operation multiple times, each with a different value for `shader_stalled_point_id`, we iterate over the different EUs and measure the relative performance of each. The output is a trace of multiple timing measurements, corresponding to the time taken by the targeted EU to draw the scene.

3.2.4 Raw Traces

Before evaluating DrawnApart, we tested whether we can visually distinguish devices. fig. 3.2 shows traces collected from two Gen 3 devices. We collect 50 traces from each device, each trace consisting of 176 measurements of 16 points.

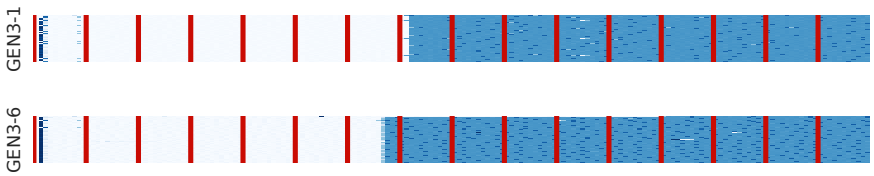


Figure 3.2: Raw traces from two different Gen 3 devices.

The measurements are divided into 16 groups of 11, where in each group we stall a different point. The color of a point indicates the rendering time, ranging from virtually 0 (white) to 90 ms (blue). Red vertical bars indicate group boundaries. As we can see, the rendering time in the first half of the traces is significantly lower than in the second half. Moreover, while there are some timing variations in the traces of the same device, the traces display patterns that are distinct between devices, allowing us to distinguish them.

3.3 Evaluation Overview

3.3.1 Motivation

We claim that our new method provides a tangible advantage over deterministic GPU-based fingerprinting. To establish this claim, we evaluate our system in a lab setting and in the wild.

In the **lab setting**, we assume the attacker can collect training traces from a set of identical machines (same hardware and software), running under identical environmental conditions. Next, the attacker is given a single trace and is tasked with identifying the machine that generated the trace. Our primary metric of evaluation in this setting is the **accuracy gain**, which measures the multiplicative gain in accuracy of a classifier that incorporates our non-deterministic method, when compared to a classifier which only uses deterministic inputs. An accuracy gain of 1 means that the classifier provides no advantage over traditional methods, while higher values show that it gives the attacker an advantage. The lab setting provides the most advantageous conditions for our classifier, for several reasons. First, existing deterministic schemes cannot tell apart identical devices, as we demonstrate experimentally, resulting in a very low base rate. Second, the attacker can tailor the attack to the particular class of devices to be discriminated, and thus choose optimal parameters for the target hardware. Third, the workload on the target machines is controlled, minimizing measurement noise. Finally, the attacker is not concerned with detectability or compatibility, and can run an experiment that takes a long time, that uses partially supported hardware features, or that is noticeable to the user.

We also evaluate our system **in the wild**. More specifically, we evaluate how our method can be applied to track devices from a set of over 2,500 machines with 1,605 distinct GPU configurations, recruited through a crowd-sourcing experiment. We first perform a standalone evaluation of our method, in the absence of additional identifying features. We then provide additional deterministic fea-

tures to the classifier, including the browser version, screen dimensions, HTTP headers, and other similar attributes. State-of-the-art fingerprinting techniques can produce unique browser fingerprints through the consideration of these signals, but these fingerprints are not ideal for tracking users since they evolve over time [140]. We therefore measure the added distinguishing power our method provides to existing browser fingerprinting schemes, with the primary metric of evaluation being the **additional tracking time** made possible through the combination of our novel technique with existing schemes.

The in-the-wild setting is more challenging. First, the technique must perform well across a large number of devices, precluding tailored attacks, and the attacker is prohibited from using any trace collection method that is overly intrusive or time-consuming. Second, the attacker's choice of machine learning pipelines is constrained. In particular, the attacker cannot use a long training phase since this does not make sense in the context of browser fingerprinting—the fingerprint should be useful at once, and not depend on the victim spending hours on the attacker's website. The attacker must also be able to accommodate new devices joining the dataset in real-time, and should not be required to spend multiple CPU hours retraining the classifier every time a new device is detected. Finally, the attacker cannot control the runtime characteristics of the machine being fingerprinted. Our method will have to be tolerant to workload variations, GPU payloads from other tabs, browser and system restarts, and so on.

In the following section, we study the lab setting to demonstrate an upper bound on our classifier's potential accuracy gain, and to investigate parameter choices and their trade-offs on accuracy, compatibility and performance. In section 3.5, we select a single set of parameters and launch a large-scale crowd-sourced experiment in the wild, showing the advantage of our method in a realistic setting.

3.3.2 Machine Learning Pipelines

We use two machine learning approaches to evaluate our fingerprinting technique. In the **lab setting**, we cast our fingerprinting problem as a conventional multinomial classification task, where the input is the trace of N rendering times, and the output is the label of the device assumed to have generated this trace. We evaluated several classical machine learning models suitable for this task, including tree-based classifiers, k-Nearest Neighbors classifiers, Linear Discriminant Analysis, and Support Vector Machines. We ultimately chose to use the Random Forest ensemble classification algorithm [157, 158], as it empirically delivered the best classification results in terms of accuracy. We did not apply any feature engineering, submitting the raw traces into the classification algorithm. To make sure we did not overfit our model, we applied a 5-fold train-test split to the data, and collected the mean accuracy reported by the folds, as well as the standard deviation among folds.

To evaluate our system **in the wild**, we needed a more elaborate pipeline for the reasons listed in section 3.3.1. Our method relies on *neural networks* and consists of several steps:

1. We preprocessed our traces by normalizing and reshaping them into matrix form.
2. We trained a convolutional neural network (*CNN*) to solve the multinomial classification task.
3. We transformed the classification network into an embedding network using the semi-hard triplet loss algorithm of Schroff *et al.* [159].

The resulting network is capable of transforming our trace into a representation called an *embedding*. Because of the way the network is designed, the Euclidean distance between two traces from the same device will be small, while the Euclidean distance between traces from different devices will be large. This allows the inference part of the classification to use the k-Nearest Neighbors classifier—given an unknown trace, measure the distance between its embedding and the embeddings of all known traces, and output the label of the embedding at the shortest distance. The simplicity of this classifier means the adversary can add new devices to the dataset simply by recording a few new traces and without retraining the entire network, a desirable property known as *few-shot learning*.

To ensure we did not overfit our in-the-wild model, we split our training dataset into two mutually exclusive parts, each with different labels, performed the evaluation on each part in isolation, and observed that the accuracies for each split were roughly the same. More details about the training process and dataset splits can be found in section 3.5.

3.4 Lab Setting

The objective of the lab setting is to discover DrawnApart's highest accuracy, and assumes that the attacker customizes the attack to the class of device and ignores aspects of detectability, compatibility or performance.

Evaluated Devices. Table 3.1 lists the devices used in the lab setting. We used 88 devices from nine distinct hardware classes, including desktops and mobile devices. The desktops include multiple generations of Intel processors, all running Windows 10, as well as a set of Apple Mac mini devices with an Apple M1 chip, running MacOS X Version 11.1. Other than the Gen 10 devices, which had discrete Nvidia GTX1650 GPUs, all desktops used integrated graphics. For each class, the devices were purchased through the same order, configured with our University's official operating system image, and located in the same temperature-controlled lab. The mobile devices include multiple generations of Samsung Galaxy devices, all sourced through the Samsung Remote Test Lab [160]. All the mobile devices were Android-based and featured Samsung Exynos CPUs and Mali GPUs.

Table 3.1: Accuracy gains achieved under lab conditions

Device Type	GPU	Device Count	Timer	Base Rate (%)	Accuracy (%)	Gain
Intel i5-3470 (Gen 3 Ivy Bridge)	Intel HD Graphics 2500	10	Onscreen	10.0	93.0±0.3	9.3
			Offscreen	10.0	36.3±1.6	3.6
Intel i5-4590 (Gen 4 Haswell)	Intel HD Graphics 4600	23	Onscreen	4.3	32.7±0.3	7.6
			Offscreen	4.3	63.7±0.6	14.7
			GPU	4.3	15.2±0.5	3.5
Intel i5-8500 (Gen 8 Coffee Lake)	Intel UHD Graphics 630	15	Onscreen	6.7	42.2±0.7	6.3
			Offscreen	6.7	55.5±0.8	8.3
			GPU	6.7	53.5±0.8	8.0
Intel i5-10500 (Gen 10 Comet Lake)	Nvidia GTX1650	10	Offscreen	10.0	70.0±0.5	7.0
			GPU	10.0	95.8±0.9	9.6
Apple Mac mini M1	Apple M1	4	Offscreen	25.0	46.9±0.4	1.9
			GPU	25.0	73.1±0.7	2.9
Samsung Galaxy S8/S8+	Mali-G71 MP20	6	Onscreen	16.7	36.7±2.7	2.2
Samsung Galaxy S9/S9+	Mali-G72 MP18	6	Onscreen	16.7	54.3±5.5	3.3
Samsung Galaxy S10e/S10/S10+	Mali-G76 MP12	8	Onscreen	12.5	54.1±1.5	4.3
Samsung Galaxy S20/S20 Ultra	Mali-G77 MP11	6	Onscreen	16.7	92.7±1.8	5.6

Comparison With Prior Fingerprinting Techniques. Before evaluating our technique, we reproduced and tested several state-of-the-art web-based fingerprinting techniques.

UniqueMachine, presented by Cao *et al.* at NDSS 2017 [161], collects a "browser fingerprint", with mutable properties such as window size and IP address, and a more permanent "computer fingerprint". The UniqueMachine website offers a demo that outputs both fingerprints as 32-character hashes. We collected the fingerprints of all of the computers in our Gen 3, Gen 4, Gen 8, and Gen 10 corpora using UniqueMachine, and confirmed that all computers in the same corpus were assigned the same computer fingerprint. Interestingly, the Gen 4 and Gen 10 PCs shared the same computer fingerprint despite having different hardware configurations.

Fingerprint JS (FPJS) is a commercial API offering "browser fingerprinting as a service". The paid-for version, called FPJS Pro, claims to provide "unparalleled accuracy, ease of use, and security" [162]. FPJS Pro outputs a 20-character hash. The website provides a demo of FPJS Pro. We collected the fingerprints of all computers in our Gen 3, Gen 4, Gen 8, and Gen 10 corpora using the demo website. In the Gen 3 dataset, all but one computer had the same fingerprint. Similarly to UniqueMachine, all of the computers in the Gen 4 and Gen 10 corpora had identical FPJS fingerprints. Finally, FPJS divided the Gen 8 corpus into three clusters: two clusters with seven computers each, and the final cluster with one computer.

Clock around the Clock, proposed by Sánchez-Rola *et al.* at CCS 2018 [163], is an alternative to GPU-based fingerprinting. This method is designed to exploit "small, but measurable, differences in the clock frequency" by measuring the precise execution times of a series of CPU-intensive operations. To calculate the fingerprint, the computer invokes the cryptographic random number generator `crypto.getRandomValues` 1,000 times for 50 different input sizes, then generates a vector of the most common timing value, or mode, for each of the input sizes. We reproduced the web-based variant of the method, and tested it on our Gen 4 corpus. We found that the modes did not contain any data useful for fingerprinting. This is likely because since July 2018 Chrome contains countermeasures designed to prevent fine-grain timing measurements, as part of the wider fallout of the Spectre attacks [164, 165, 166, 167]. All our measurements returned either zero or five microseconds (with some added randomness). We conclude that, currently, the method presented by Sánchez-Rola *et al.* is not practical.

3.4.1 Tuning the Trace Parameters

We search for the parameter settings that provide the optimal accuracy gain for the different hardware configurations.

Stall Function Operator Selection. Each model and generation of GPU has a different micro architecture. For example, the third-generation Intel integrated GPU has a single arbiter, which dispatches tasks to all EUs, while fourth-generation GPUs adopt a hierarchical micro-architecture with multiple arbiters. Intel GPUs also have *Advanced Math (AM) Units*, which are tasked

with executing less common operations such as trigonometric operations. The amount and location of these AM units differs among GPU generations, and even within different GPU types from the same generation. The design of GPUs by Nvidia, ARM and Apple is obviously different as well. We hypothesize that, due to these differences, the accuracy gain provided by our method will vary, depending both on the choice of stall functions and target hardware. To test this, we evaluated a representative set of operators, including trigonometric operations, logical bit-wise operations, and general floating-point operations.

We report the complete evaluation of the selected set of operators for 600 traces, 20 points, and 5 iterations per point in the online setting, for the Gen 3 (table 3.2), Gen 4 (table 3.3), and Gen 8 (table 3.4) datasets.

Table 3.2: Evaluation for the Gen 3 dataset, depending on the operators. The baseline is 10%

Operator	Accuracy	Median time (ms)
`mul`	$93.5\% \pm 0.7\%$	3,097
`sinh`	$89.5\% \pm 1.4\%$	8,757
`abs`	$88.4\% \pm 1.0\%$	4,532
`pow`	$87.5\% \pm 0.3\%$	4,663
`log`	$87.1\% \pm 1.2\%$	9,839
`exp2`	$87.0\% \pm 0.6\%$	7,532
`shl`	$86.3\% \pm 1.7\%$	6,799
`atanh`	$81.8\% \pm 1.4\%$	11,184
`inversesqrt`	$80.1\% \pm 0.7\%$	6,799
`trunc`	$67.1\% \pm 1.4\%$	1,667

Table 3.3: Evaluation for the Gen 4 dataset, depending on the operators. The baseline is 4%

Operator	Accuracy	Median time (ms)
`mul`	$32.7\% \pm 0.3\%$	6,361
`abs`	$29.3\% \pm 0.5\%$	3,295
`shl`	$28.7\% \pm 0.4\%$	6,483
`inversesqrt`	$28.2\% \pm 0.7\%$	6,485
`exp2`	$27.8\% \pm 1.0\%$	6,528
`trunc`	$26.6\% \pm 1.1\%$	3,161
`log`	$25.3\% \pm 1.0\%$	7,673
`pow`	$23.3\% \pm 0.6\%$	9,370
`sinh`	$19.5\% \pm 0.5\%$	8,953
`atanh`	$19.1\% \pm 0.6\%$	10,099

Table 3.4: Evaluation for the Gen 8 dataset, depending on the operators. The baseline is 6%

Operator	Accuracy	Median time (ms)
`exp2`	$43.6\% \pm 1.2\%$	3,172
`inversesqrt`	$39.4\% \pm 0.9\%$	3,181
`pow`	$36.6\% \pm 1.4\%$	4,698
`log`	$33.6\% \pm 0.5\%$	3,299
`sinh`	$32.4\% \pm 0.9\%$	4,569
`abs`	$30.9\% \pm 0.6\%$	3,174
`mul`	$30.9\% \pm 1.0\%$	3,173
`atanh`	$30.6\% \pm 0.5\%$	5,935
`trunc`	$28.7\% \pm 0.6\%$	3,174
`shl`	$26.9\% \pm 1.1\%$	3,172

Timing Measurement Method. Scene rendering is performed in the GPU context, which is asynchronous to the CPU context. Simply measuring the time it takes the CPU to execute the draw operation, for example by calling `performance.now()` immediately before and after the call, does not provide any usable insight about the GPU. We therefore considered three measurement methods that are capable of measuring the actual drawing time of the GPU.

In the **onscreen** method, we render the scene to a standard HTML canvas element and then call `Window.requestAnimationFrame`. This function is passed a callback function that is called after the rendering is complete. Timing information is then extracted from within the callback. The onscreen method is the most compatible of those we evaluated, but browsers do not call `requestAnimationFrame` at a rate higher than the browser's maximum frame rate, which is typically 60 Hz. Thus, using this method requires that each iteration of our rendering operation take at least 16 ms to provide us with useful information. Even though the canvas element is on screen, it can be made zero-size or invisible via styling, making the fingerprinting operation invisible to the user. Collecting the fingerprint does cause a noticeable slowdown for the user since it runs in the browser's main context.

In the **offscreen** method we use a worker thread and render the scene to an `OffscreenCanvas` object. This does not affect the user's main context and does not slow down the user. After rendering the scene, we call the `convertToBlob` method of the `OffscreenCanvas`, causing it to execute all instructions currently in the WebGL pipeline, and ultimately return a binary object representing the image contained in the canvas. We measure the time it takes to execute this command. Since there is no frame rate limit in this setting, each iteration of the rendering operation can take less time, allowing us to use more iterations. At the time of writing, `OffscreenCanvas` is supported on Chrome browsers, hidden behind a flag on Firefox, and partially supported in the Technical Preview build of Safari.

The **GPU** method is the third method we evaluate. It is a modification of the offscreen method that does not measure timing on the CPU side. Instead, the WebGL disjoint timer query method is used to directly measure the duration of a set of graphics commands on the GPU side. To perform this measurement, we call `beginQuery`, issue the drawing operations, and call `endQuery`. Using `getQueryParameter`, we retrieve the elapsed time on the GPU side. This disjoint timer query command was previously used for side-channel attacks by Frigo *et al.* in their work in IEEE S&P 2018 [168]. As a result, support for this timer was disabled in Chrome version 65. However, with the introduction of Site Isolation [169], it was deemed safe to be re-enabled in Chrome version 70 [170]. In contrast to CPU-side timers, whose resolutions have been severely reduced to a few microseconds with jitter to mitigate against transient execution attacks [171], the GPU-side timer offers microsecond resolution with no jitter even on the most modern versions of Chrome [172]. This GPU-based timer thus has the potential to be the most accurate and the least sensitive to activity on the CPU side. On the other hand, its accuracy varies dramatically between different GPU architectures, and it is not supported by the commonly used Google SwiftShader renderer.

Number Of Points To Render. Our fingerprinting scheme relies on multiple iterations of a drawing command, where each iteration exercises a certain subset of the EUs while leaving the other EUs idle. The number of iterations and the time each iteration takes to run will determine the total execution time. However, it is reasonable to assume that capturing more data will provide better accuracy, and that relatively long workloads will mitigate the impact of the low-resolution timers available through JavaScript. We ran two experiments to capture this trade-off. The first was run in the onscreen setting, using the Gen 3 corpus. The frame rate requirement of the onscreen setting limits each iteration to at least 16 ms, as explained above. The second experiment was run in the offscreen setting using the Gen 4 corpus. This setting allowed us to use much shorter workloads and to increase the number of iterations that can be run in a reasonable time period. Thus, instead of assigning the stall function for each point only once per iteration, we tried all 2^n possible subsets of the set of points, allowing us to measure the *contention* between EUs, as well as their individual speeds.

3.4.2 Results

Table 3.1 summarizes the accuracy gains obtained in the lab setting using different timing methods. The mobile devices were evaluated using the onscreen method only due to limited access to those devices. Gen 3 and Gen 4 are not evaluated using the GPU timer method since their hardware does not support it. All devices within each hardware class were sampled the same amount of times. We observed that our Random Forest-based classifier approaches peak accuracy as the size of the training data set approaches 500 traces per label. As the table shows, our scheme delivered significant accuracy gains, well above the base rate, in all scenarios, both for desktop and mobile devices. The parameter choices, however, did affect the performance of our scheme.

Effect Of Stall Function. As expected, each of the operators we evaluated performed differently on the different hardware targets. Specifically, in the onscreen setting, the `mul` operator delivered the best accuracy gains for the Gen 3 and Gen 4 corpora, while `exp2` was the best performer for the Gen 8 corpora, as described in more detail in section 3.4.1. The different mobile device corpora, which were also evaluated in the onscreen setting, also had different optimal operators: `pow` for Galaxy S8/S8+ and Galaxy S9/S9+, `atanh` for Galaxy S10e/S10/S10+ and `mul` for Galaxy S20/S20+/S20 Ultra.

In the offscreen setting, the `sinh` operator was consistently the best performer for the Gen 4 and Gen 8 corpora, while `mul` was better than `sinh` for the Gen 10 corpora. We hypothesize that since the offscreen setting allowed us to trigger multiple execution units at the same time, and the amount of advanced math units that handle trigonometric operations is lower than the amount of EUs, the conflicts and race conditions that arise inside the GPU gave this operator additional discriminating power.

Effect Of Timing Measurement Method. As stated above, the offscreen method allowed us to execute more iterations than the onscreen method, allowing us to capture data about EU contention, as well as on the timing of individual EUs. We were also interested in comparing the relative performance of the offscreen method, which measured time on the CPU side, and the GPU method, which used disjoint timer queries to measure performance on the GPU side. We hypothesizes that the GPU method would be superior to the offscreen method, since the GPU-side timer has higher accuracy than the CPU-side timer, and is not affected by the timing jitter introduced by inter-process communications (IPC) between the GPU and the CPU. In practice, we discovered that this is not always the case. As shown in table 3.1, the GPU timer is better than the CPU timer for the Intel Gen 10 and Apple M1 corpora, has equivalent accuracy to the CPU timer on the Gen 8 corpus, and is actually less accurate than the CPU timer on the Intel Gen 4 corpus. To make matters worse, the disjoint timer query WebGL extension is not supported on several popular WebGL stacks, most significantly the software-based Google SwiftShader. Thus, the GPU-based timer is not appropriate for use in a large-scale experiment where the hardware configuration is not known beforehand.

Accuracy vs. Capture Time. Figure 3.3 shows the accuracy gain as a function of trace capture time, both for the Gen 3 corpus using the onscreen collection method, and for the Gen 4 using the offscreen collection method. As the Figure shows, the accuracy gain of both methods approaches its optimal point when samples are collected for around 2 seconds. This is reached after about 80 iterations in the onscreen method and 1024 iterations in the offscreen method.

Swapping Hardware. To reinforce our claim that the classification results are due to differences in the behavior of the GPUs, and not due to some residual differences among the computers, we selected two Gen 3 computers, physically swapped their hard drives, and re-ran the fingerprinting classifier. As expected, the fingerprinting classifier was not misled by the hard disk transplant, and was still able to label each of the two computers according to their CPU. Next, we returned the hard drives to their original locations, and physically swapped the

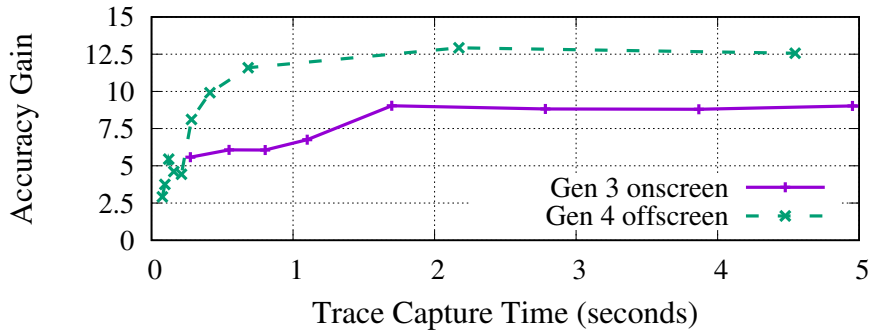


Figure 3.3: Accuracy gain as a function of trace capture time

CPUs with integrated graphics of the two systems. As expected, the classifier followed the transplanted CPU, even though all other hardware was unmodified.

3.4.3 Evaluation on Additional Browsers.

We collected and evaluated traces from 16 devices from the Gen 4 corpora using multiple additional browsers: Brave browser [173] (version 81.0.4044.113), Edge [174] (version 96.0.1054.43), Opera [175] (version 82.0.4227.23) and Yandex browser [176] (version 21.11.3.927), all using the *offscreen* method. The accuracy showed a significant improvement over the base rate, which lies at 6.25%, with Edge, Brave, Opera and Yandex, delivering accuracies of 34.6±0.6%, 31.0±0.3%, 31.6±0.7%, and 31.1±0.3%, respectively.

We evaluated the stability of DrawnApart over 21 devices of the Gen 4 corpora for an extended period of time. We collect data for both Chrome and Firefox. For Chrome, we use the *onscreen* and *offscreen* methods. For Firefox, which does not currently support the *offscreen* method, we are limited to the *onscreen* method. We also chose to stall the EU for twice as many operations under Firefox, compared to Chrome, to account for the lower timer resolution found in Firefox.

For 24 days, we repeatedly launched the browser, collected traces for 20 minutes using the *offscreen* method and for 40 minutes using the *onscreen* method, then quit the browser and idled for 4 hours. The first 4 cycles were used to train the Random Forest classifier, while the remaining cycles over the experiment's 24 days were used to evaluate its performance. The results are summarized in Figure 3.4 and show the accuracy to be above the base rate for each point in time. We observed that the *offscreen* method yields slightly higher accuracy than *onscreen*, and that the accuracy of both methods on Chrome slightly decay over time, while the accuracy of the *onscreen* method on Firefox remains stable. Finally, the accuracy in this experiment is lower compared to the results reported in Table 3.1. It is possibly due to repeatedly restarting the browser over the course of the experiment, as we discuss in Section 3.6.3.

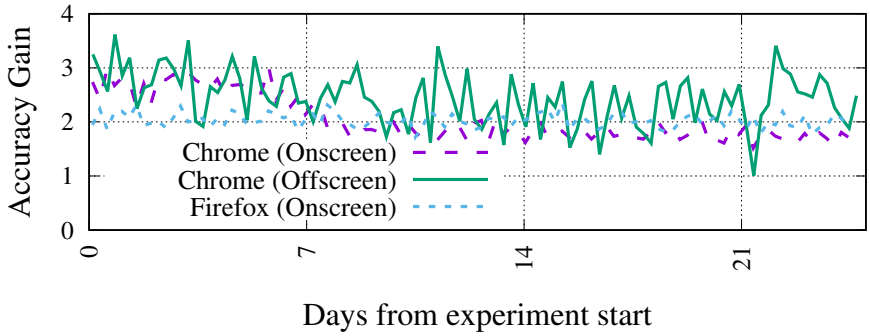


Figure 3.4: Additional Browsers – Lab Evaluation

3.4.4 Summary

Our results show that DrawnApart can tell apart identical computers in a controlled lab setting. Our next objective was to a realistic setting, in which the attacker has less control over the devices to be fingerprinted. We did so by first evaluating DrawnApart in a standalone setting, and then integrating it with a state-of-the-art browser fingerprinting algorithm.

3.5 In-the-Wild Setting

Performing browser fingerprinting in the wild presents different challenges compared to what we experienced with the lab setting:

1. The lab evaluation assumed a closed list of devices. In the real world, new devices can be added at any time during the collection period, but we cannot re-train the model whenever it happens.
2. The lab evaluation assumed we had a long time to collect data and train over the devices. In the real world, we do not have unlimited access to a device so the collection of data must be fast.
3. Finally, the lab evaluation assumed the devices were idle and in a controlled environment. In the real world, we have to contend with variable computing loads, restarts, and updates to both the browser and the operating system.

In order to understand the potential impact of DrawnApart in the real world, we collected $370,392$ traces from $2,550$ devices over 7 months and performed the two following evaluations:

- **Standalone evaluation**: Considering only DrawnApart traces without any other information, we aim to see how our method performs at reidentifying a device among others. In section 3.5.2, we propose a one-shot learning pipeline whose aim is to match a new trace with another known trace present in our dataset.
- **Tracking over time**: Browser fingerprints evolve [152]. Vastel *et al.* developed two algorithms to track evolutions and link fingerprints that be-

long to the same device [177]. In section 3.5.4, we show how DrawnApart can improve the FP-Stalker algorithms, which are the current state-of-the-art tracking algorithms, by increasing the duration users can be tracked. Our main metric to evaluate the gain of our technique will be the **median tracking time**. Contrary to the standalone evaluation, we use all the attributes listed in appendix A.1 as well as the DrawnApart traces.

3.5.1 Dataset constitution

Large-scale Experiment. To show DrawnApart's practical advantages over traditional deterministic fingerprinting methods as used in FP-Stalker, we launched a large-scale experiment with diverse hardware and software. We integrated our DrawnApart technique into the Chrome browser extension from the AmIUnique crowd-source experiment [155]. The extension periodically collects the browser fingerprints of thousands of volunteers, allowing us to track their evolution.

DrawnApart Collection Parameters. The crowd-sourced experiment constrained our choices. Most importantly, we wanted to be as non-intrusive as possible, as to not cause any user-perceivable slowdowns. In addition, we wanted to be compatible with various rendering stacks we encounter in the wild. Finally, we were interested in selecting a stall function that discriminates a wide variety of hardware. With these constraints, we selected the **offscreen** timing method, which is supported by all desktop versions of Chrome. The onscreen method was not selected as it causes slowdowns, and the GPU method was not selected since it is not supported by the Google SwiftShader renderer. We chose the `sinh` stall function operator, which provided good performance during our tests. We render all possible subsets of 10 points in each trace, for a total of $2^{10} = 1,024$ iterations per trace. This fingerprint takes a median time of 1.6 seconds to run. It is collected by the extension using a worker thread, without affecting the user's interactions with the browser. To increase our trace count, we repeated each collection seven times, for a median total run time of approximately 12 seconds. We collected the traces every four hours.

Dataset Preparation. Our dataset contains $370,392$ fingerprints from $2,550$ unique devices. In each fingerprint, we collect the attributes listed in appendix A.1, together with 7 DrawnApart traces. We identify devices with the same GPU by looking at the *WebGL renderer string* property. Over 90% of the devices shared a renderer string with at least one additional device. The largest observed group with same renderer string consisted of 534 unique devices.

We split our dataset into three subsets, divided by measurement time: 1MP contains $109,375$ samples collected between 3-Jan-2021 and 7-Feb-2021, 2MP contains $46,293$ samples collected between 7-Feb-2021 and 31-Mar-2021, and 3MP contains $214,724$ samples collected from 3-May-2021 to 8-Jul-2021. We randomly choose 65% of the devices in 1MP that have more than 28 samples, and refer to this subset as $\mathsf{1MP}_{65}$. The rest of 1MP will be referred to as $\mathsf{1MP}_{rest}$ The limit of 28 samples, or 196 DrawnApart traces, was chosen to make sure the neural network will generalize well, by preventing it from overfitting on

a small amount of traces of a specific device. We normalized each trace and reshaped a vector of length 1024 into a 32x32 matrix.

3.5.2 Standalone evaluation

Before integrating our model with FP-Stalker, we first evaluate it in isolation using only DrawnApart traces and ignoring the other attributes. In contrast to the classical ML model used in the lab setting, we used a neural network pipeline for the in-the-wild setting. The ultimate goal of the pipeline is to generate quality embeddings in Euclidean space, which express the distance between traces. We begin the process by creating a *Convolutional Neural Network* (CNN)-based multinomial classifier. The structure we selected for the classifier is inspired by Picek [178], and includes N convolution blocks followed by a flatten layer, a dense layer, another L2-normalized dense layer without activation, and concluding with a fully connected layer with softmax activation. Each convolution block contains a convolution layer, a dropout layer, and an average pooling layer. We used scikit-optimize's Bayesian optimization [179] to search for the best parameters, using 80% of the traces in $\mathsf{1MP}_{65}$ for training, and the remainder of $\mathsf{1MP}_{65}$ for validation. The chosen hyperparameters are listed in Table 3.5. The parameter search took 48 hours on a server with four NVIDIA GEFORCE RTX 2080 Ti GPUs, two Intel Xeon Silver 4110 CPUs, and 128 GiB of RAM. The run yielded 79 valid neural networks. The best network achieved a training accuracy of 35.57% and a validation accuracy of 33.82%.

Table 3.5: Hyperparameters for the CNN classifier

Hyperparameter	**Value**	**Space**
Embedding size	256	32–256
Number of convolution blocks	3	1–10
Batch size	32	32–1024
Convolution filter size	128	8–128
Convolution kernel size	4	2–5
Dropout rate	0.119510	0–0.5
Activation	relu	relu, sigmoid

Semi-Hard Triplet Loss Model. The next step in our ML pipeline is the transformation of the multinomial classifier into an embedding, using the triplet loss method. Triplet loss minimizes the distance between an anchor and a positive, both of which have the same label, and maximizes the distance between the anchor and a negative of a different label. Semi-hard triplet loss means that we only use triplets that have a negative that is farther from the anchor than the positive, but still produces a positive loss [159]. We took our trained classification model, removed its last layer, and trained it again for 30 epochs on the same dataset as before, this time with a bigger batch size of 1024 preprocessed traces and with semi-hard triplet loss. Batch size is important to the triplet mining process since we need sufficient examples in the batch to find enough semi-hard triplets. We took the weights of the epoch that yielded a model with the best accuracy using a 1-Nearest Neighbor classifier. The end-product of this process is a model that accepts preprocessed DrawnApart traces as input and produces embeddings in a Euclidean space. Labels are not involved in this process—we can take any DrawnApart trace, even from a

device that the model was not trained on, feed it into the triplet loss model, and get Euclidean space embeddings. We note that we optimized for the accuracy of the classification model, instead of the 1-Nearest Neighbor, to reduce the running time of our parameter search.

Evaluating The Classifier. The use of embeddings mandates using a k-Nearest Neighbors classifier for analyzing the outputs of the network. Our metric for evaluation is the top-k accuracy, which stands for the probability that the correct answer is one of the k nearest neighbors of the selected trace, for $k = 1$, 5, and 10, according to the distance metric output by the model.

Base Rate Calculation. The accuracy of a classifier should be compared to the *base rate* obtained by a naive classifier with no access to the features. In the case of a classical learning problem, the naive classifier can observe the training data and learn the apriori probabilities of each label. Then, to get the best accuracy, this naive classifier will output the label of the most commonly observed device, or the n most commonly observed devices for a top-n setting. The base rate in that case is therefore the cumulative proportion of these devices in the dataset. In the case of a k-shot learning problem, the classifier does not know the apriori probabilities of each label, since it gets an equal amount of training data for each label. The naive classifier in this case will just output a random label, or n random labels for a top-n setting. The base rate in that case is only $n * (\#devices)^{-1}$.

Train-Test Split Evaluation. We evaluated our model in two ways: random train-test split, and k-shot learning. In the train-test split evaluation, we randomly split each of the $\mathsf{1MP}_{65}$, $\mathsf{1MP}_{rest}$ and 2MP datasets into two parts, using 80% for memorizing and 20% for testing. We first used $\mathsf{1MP}_{65}$ for evaluation. On this subset, the base rate is 1.00% for top-1 accuracy, 3.51% for top-5 accuracy and 6.15% for top-10 accuracy. To show that our network can generalize and work on traces it has never seen before, we next considered the performance of the network on $\mathsf{1MP}_{rest}$. On this subset, the base rate is 1.22% for top-1 accuracy, 4.42% for top-5 accuracy and 7.2% for top-10 accuracy. To show that our network generalizes to more devices and new traces, we evaluate it on 2MP. 2MP contains devices from 1MP, meaning that the neural network was trained on some of the devices in 2MP, but not all of them, but it was never trained on any traces from 2MP. On this subset, the base rate is 0.64% for top-1 accuracy, 2.78% for top-5 accuracy and 4.38% for top-10 accuracy. The results in table 3.6 demonstrate that our model accuracies are significantly better than the base rate for all of the three datasets. The accuracies on $\mathsf{1MP}_{65}$ and $\mathsf{1MP}_{rest}$ datasets are roughly the same, showing the model responds well to new devices. The small drop in the accuracy of 2MP despite a base rate of approximately half the other datasets, the addition of more devices and new traces and being collected at a later date, shows the model has generalized well.

k-shot Learning Evaluation. The k-shot learning evaluation was performed on the 2MP dataset. We chose 2MP to evaluate k-shot learning because we used the traces from $\mathsf{1MP}_{65}$ to train our triplet loss model, which would bias the results. While some of the devices in this subset also appear in 1MP, none of the traces in 2MP were used to train or validate the neural network. In the memorizing phase, we memorize the first k collections ($k \times 7$ DrawnApart

Table 3.6: Standalone Performance of DrawnApart In the Wild Using The Random Split (RS) and k-shot Methods

Evaluation Method (Dataset)	Accuracy (*Base rate*) **Top-1**	**Top-5**	**Top-10**
RS ($1MP_{65}$)	28.88% (*1.00%*)	56.36% (*3.51%*)	68.70% (*6.15%*)
RS ($1MP_{rest}$)	28.28% (*1.22%*)	55.09% (*4.42%*)	67.15% (*7.20%*)
RS (2MP)	23.33% (*0.64%*)	47.23% (*2.78%*)	58.83% (*4.38%*)
1-Shot (2MP)	5.44% (*0.05%*)	14.10% (*0.26%*)	19.95% (*0.51%*)
5-Shot (2MP)	7.11% (*0.05%*)	19.34% (*0.26%*)	26.75% (*0.51%*)
10-Shot (2MP)	9.22% (*0.05%*)	22.77% (*0.26%*)	31.09% (*0.51%*)

traces) of each device in 2MP. The rest of the traces of 2MP are used in the testing phase, again using a k-Nearest Neighbors classifier. This is an evaluation that is close to real-world use. An attacker would like to identify users with as few collections as possible. This evaluation is harder than the previous one due to the small amount of data available for the memorizing phase. In addition, the time difference between 1MP and 2MP requires the network to deal with concept drift. As mentioned above, the base rate in this setting is very small, because the attacker cannot learn anything about the distribution of the devices in the test set. The results can be found in table 3.6. As expected, they show a decrease in accuracy compared to the evaluation using random split, but our model still delivers significant accuracy beyond the base rate. We thus conclude that DrawnApart can be used for few-shot learning.

We leave the 3MP dataset to be used in the evaluation process of FP-Stalker to test the model on a truly unseen dataset that reproduces in-the-wild conditions.

Visualizing Euclidean Distances. To visualize the performance of our few-shot learning pipeline, we computed the Euclidean distances between pairs randomly sampled from 2MP from the three following populations: Embeddings from the same device, embeddings from different devices that share the same renderer string, and finally embeddings from different devices with different renderer strings. To eliminate correlations between traces in the same collection, we used only the first trace in the collections that we sampled from. It means that we measured the distance between traces from different collections only. fig. 3.5 presents the probability density of the different distributions. As the figure shows, embeddings from the same device get a lower Euclidean distance compared to embeddings from different devices, even if the device has the same GPU. Of interest is that embeddings from different devices that share the same renderer string have a lower Euclidean distance compared to different devices that do not share the same renderer string. This confirms that DrawnApart indeed fingerprints the GPU stack or an element correlated with the GPU stack. We can also observe that if two traces have a Euclidean distance of less than 0.65, we can be almost certain that both traces came from the same device. This is a strong property, we use it in the next section to improve FP-Stalker.

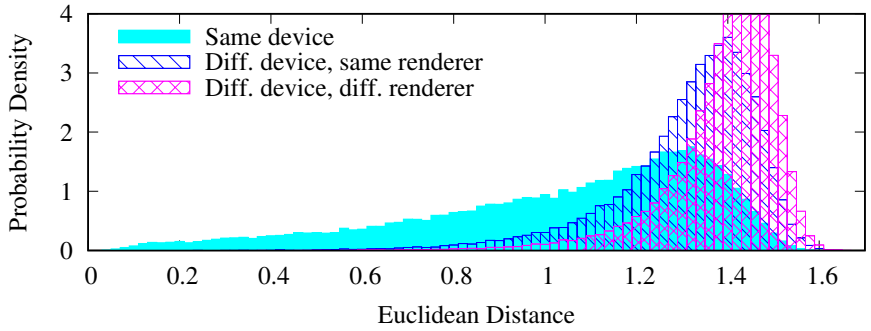


Figure 3.5: Performance of the DrawnApart embedding function. A Euclidean distance below 0.65 indicates that the traces are likely to be from the same device.

3.5.3 Evaluation on additional browsers in the wild.

While approximately 93.8% of the traces found in our in-the-wild dataset 2MP come from users running the Google Chrome browser, some users submitted traces using other Chromium-based browsers. We isolated non-Chrome users by filtering the traces according to their *user-agent*, and analyzed the effectiveness of our standalone machine learning pipeline on these browsers as well. The non-Chrome traces came from users running Edge, Opera and Yandex, which represented 5%, 0.7% and 0.5% of the traces respectively. We run the evaluation pipeline described in section 3.5.2 for each browser, independently. Our results show that the standalone pipeline's accuracy for Edge, Opera and Yandex is 52.6%, 79.3%, and 89.7%, respectively. The smaller amount of traces in this subset of the data results in a higher base rate when compared to the entire 2MP dataset—3% for Edge, 17.9% for Opera, and 27.6% for Yandex. These results, with the lab setting results, indicate that our fingerprinting technique identifies browsers from multiple vendors. More details can be found in appendix A.2.

Summary. The results of the standalone evaluation, as summarized in Table 3.6, show a significant improvement over the base rate, demonstrating that DrawnApart is effective on its own. However, it can be observed that the classifier's effectiveness is significantly reduced in the k-shot model, where the attacker has a limited trace budget to be used for training. Putting these numbers into context is important. In the world of browser fingerprinting, no single attribute differentiates all devices. While some attributes are more discriminating than others, it is their combination that is key to differentiating one device from another. The standalone evaluation of DrawnApart shows that our method has the potential to significantly contribute to fingerprinting accuracy. In the following subsection, we empirically measure this contribution by using our method in conjunction with additional fingerprinting attributes.

3.5.4 Integrating DrawnApart with FP-Stalker

FP-Stalker is the state-of-the-art fingerprint linking algorithm [140]. In this section, we show that DrawnApart can be used to improve the state-of-the-art.

Hybrid Algorithm. FP-Stalker has two distinct algorithms: one entirely rule-based, while the other combines rule-based constraints and machine-learning. Vastel *et al.* demonstrated that their hybrid variant of yielded better results on their dataset, but was slower than its rule-based counterpart. As we are trying to prove the effectiveness of DrawnApart in a real-world scenario, we chose to implement and optimize the hybrid FP-Stalker algorithm, regardless of its speed.

FP-Stalker consists in:

1. a preprocessing step that discards fingerprints that contain inconsistencies or have been spoofed and cannot be normally found in the wild,
2. a training phase, in which the Random Forest algorithm is trained on a balanced dataset,
3. an inference phase, in which the trained model, combined with rules, compares incoming fingerprints to a pool of previously classified fingerprints and attempts to link them.

The linking algorithm is presented in Algorithm 1.

Improving The Algorithm. As mentioned in section 3.5.2, the output of the embedding network consists of 256 L2-normalized points that allow us to use a Euclidean distance to compute the similarity between embeddings. fig. 3.5 shows that the Euclidean distance is efficient, to an extent, in differentiating devices. Based on the results obtained in section 3.5.2, which show that DrawnApart can correctly classify devices with an acceptable accuracy, we decided to introduce the use of the generated embeddings as a complement to the machine-learning side of FP-Stalker. We note that the results of our nude FP-Stalker cannot be fully compared to the results obtained by Vastel *et al.* for two main reasons:

1. Their dataset spans for longer than the dataset we use in our experiments.
2. Flash-related attributes no longer exist,[180], negatively impacting FP-Stalker's effectiveness.

Integrating DrawnApart as a complement to FP-Stalker's machine-learning model is motivated by the fact that FP-Stalker uses a series of conditions on the output of the Random Forest that makes its decisions too restrictive. FP-Stalker's original code includes a function to optimize the threshold used by the Random Forest, which we adapted and ran on our dataset. The resulting threshold yielded similar results, consequently comforting our observation that the rules associated to the output of the Random Forest are too restrictive, and discard too many fingerprints coming from the same browser instance. On the other side, fig. 3.5 shows that even though the Euclidean distances can be used to efficiently differentiate devices with a relatively low threshold, its usage alone may yield an unacceptable rate of false linkages due to a little

Function *FingerprintMatching (F, f_u, λ, ϵ)*
  for $f_k \in F$ **do**
    if *FingerPrintHasDifferences(f_k, f_u, rules)* **then** $F_{ksub} \leftarrow$ exact $\cup < f_k >$;
    else
      exact $\leftarrow$ exact $\cup$ ¡f_k¿
    end
  end
  if $|exact| > 0$ **then**
    if *SameIds(exact)* **then return** *exact[0].id* ;
    else return *GenerateNewId()* ;
  end
  for $f_k \in F_{ksub}$ **do**
    cosine_sim $\leftarrow$ GetSimilarity($f_u.avg\_embedding$, $f_k.avg\_embedding$);
    if $cosine\_sim > \epsilon$ **then**
      return $f_k.id$
    end
    $< x_1, x_2, .., x_m >$ = FeatureVector(f_u, f_k);
    p $\leftarrow$ P($f_u.id = f_k.id \mid < x_1, x_2, .., x_m >$)
    if $p \geq \lambda$ **then**
      candidates $\leftarrow$ candidates $\cup < f_k, p >$
    end
  end
  if $|candidates| > 0$ **then**
    if $|GetRankAndFilter(candidates)| > 0$ **then return** *candidates[0].id* ;
  end
  return *GenerateNewId()*

Algorithm 1: Hybrid matching algorithm with the DrawnApart addition highlighted in red

percentage of different devices having low Euclidean distances. To use DrawnApart embeddings in FP-Stalker, we average the seven embeddings that are collected with each fingerprint and we output an average embedding. We used the previously generated averaged embeddings to compute the cosine similarity of the two compared fingerprints. The resulting similarity is compared to a threshold we chose based on an analysis on the train dataset. This process is explained in the next paragraphs. If the similarity of the two embeddings is above the chosen threshold, we classify the fingerprint as similar to the one being compared without further steps. The algorithm with the DrawnApart additions, is available in appendix A.1.

Choosing The Epsilon Threshold. We chose the threshold by performing an analysis over similar and different devices on the train dataset. We generate an equally balanced dataset from the training set comprising the cosine similarity of similar devices and different devices, and compare different percentiles of the distance of each group. As opposed to the Euclidean distance used in fig. 3.5,

Table 3.7: Average tracking time by collection period

Collection Period	Tracking duration in days		Improvement
	Nude FPS	FPS+DA	
2 days	17	26	+52.94%
3 days	17.25	25.5	+47.82%
4 days	17	28	+64.70%
5 days	17.5	27.5	+57.14%
6 days	18	30	+66.66%
7 days	17.5	28	+60.00%

was chose the cosine similarity for FP-Stalker because it is bounded by a more natural interval of $[-1; 1]$. Our experiments showed that our threshold on the cosine similarity yielded better results than our Euclidean distance threshold. Following our analysis, we noticed that the 5th-percentile of similar devices are all comprised below a similarity of 0.10. Consequently, we chose a threshold of 0.15 in our experiments to account for a safety margin.

Results. We executed our revisited FP-Stalker with its DrawnApart addition on the dataset described in section 3.5.1. We first trained the Random Forest model on fingerprints in the 1MP subset. We then executed the lambda optimization in order to run FP-Stalker with its optimal parameters, as required by the original paper. Finally, we executed the inference phase on 3MP, which is unseen by the training phase of both FP-Stalker and the embedding's network. We execute both FP-Stalker without our contribution, and our revisited version with DrawnApart, on the same dataset for collection periods ranging from two to seven days. table 3.7 presents the average tracking duration obtained for each collection period, with a top improvement of 66.66% compared to the original FP-Stalker on a collection period of six days. fig. 3.6 presents the average tracking duration with a collection period of seven days, as presented in the original paper, which represents tracking a user who visits a website once a week. As the figure shows, adding DrawnApart to FP-Stalker increases the tracking time, raising the median average tracking time by **10.5 days**, from **17.5 days** to **28 days**. This is a substantial improvement to stateless tracking, obtained through the use of our new fingerprinting method, without making any changes to the permission model or runtime assumptions of the browser fingerprinting adversary. We believe it raises practical concerns about the privacy of users being subjected to fingerprinting.

3.6 Discussion

3.6.1 Ethical Concerns

We integrated our fingerprinting algorithm into the Chrome browser extension from the AmIUnique crowd-sourced experiment in January 2021. On the installation page, users are informed of its purpose and of the data that is collected. To safeguard users' privacy, collected traces are only associated with a random identifier created when the extension is installed, and participants can delete all

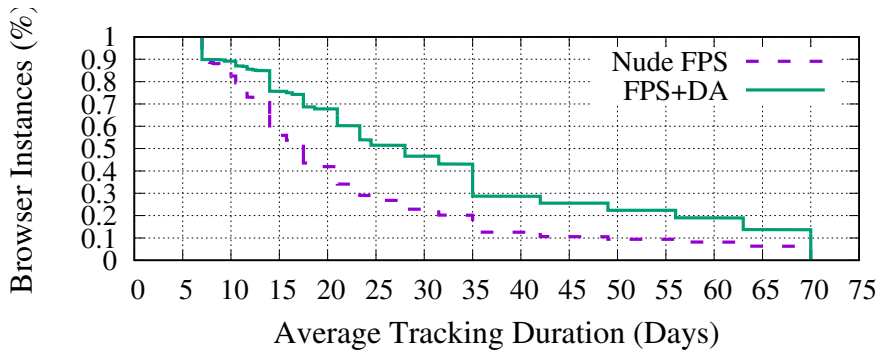


Figure 3.6: Differences in Average Tracking Time between FP-Stalker (Nude FPS) and FP-Stalker with DrawnApart (FPS+DA)

their data by submitting their extension ID. Out of an abundance of caution, we decided not to publish the weights of the triplet loss model trained on these users, since it can enable an attacker to track these users. The extension and the handling of collected data conform to the IRB recommendations we received.

3.6.2 Fingerprinting countermeasures

Countermeasures can be divided into three groups.

Blocking Scripts. Filter lists block resources known to be a threat to user privacy. This is the case of Brave's Shield mechanism [181] and extensions, such as Ghostery [182] or Privacy Badger [183]. However, filter lists against trackers and fingerprinting have been shown to lack exhaustiveness [184, 185].

API Blocking. Tor Browser, by default, and Firefox, with specific configuration, prevent web pages from reading out the contents of the canvas for privacy reasons. Our technique does not examine the canvas content, but rather measures the time required to draw different graphics primitives. Snyder *et al.* [186] consider the WebGL specification a "low-benefit, high-cost standard", which is required by less than 1% of the Alexa Top 10k websites. This may lead some people to consider the extreme option of completely blocking WebGL, as possible way of preventing GPU fingerprinting. Disabling WebGL, however, would have a non-negligible usability cost, especially considering that many major websites rely on it, including Google Maps, Microsoft Office Online, Amazon and IKEA. As a form of compromise, we note that Tor Browser currently runs WebGL in a "minimum capability mode", which allows some WebGL functionality while preventing access to the `ANGLE_instanced_arrays` API used by our attack.

Changing Attribute Values. Defenses can change an attribute value either to make it similar with common values shared by a large proportion of users, or to add noise to it. For example, Tor Browser unifies the values of many attributes for all users so that their fingerprint is identical, and some browser extensions add noise to rendered canvas images [177]. Wu *et al.* [187] introduced a countermeasure that eliminates the differences in floating point operations during the rendering process to eliminate the differences in the rendering composition

of WebGL. Blurring defenses on canvas and WebGL focus on changing values. Our technique does not directly rely on the differences in images in a rendering process, and therefore is not affected by the countermeasure of Wu *et al.* [187].

There are three elements that are crucial to our fingerprinting technique: the ability to issue drawing operations in parallel. The entire graphics stack tendency to deterministically choose which EU will render each vertex. And the ability to measure the time it takes to render. Disrupting any of these elements could affect the accuracy of our technique.

Preventing Parallel Execution. To block our method, graphics stack could limit each web page to a single EU, or disable hardware-accelerated rendering altogether and use a deterministic software-only pipeline [187]. However, this would severely affect usability and responsiveness, because WebGL is built around massive parallelism. Existing graphics APIs do not also support partitioning execution to a subset of EUs at the moment.

Preventing Deterministic Dispatching. Adding a randomization step to the GPU's dispatcher would make it impossible for the web page to choose which EU receives which vertex. Assuming the dispatcher still attempts to fill up all available EUs, the effect on performance can be minimized. We note that this countermeasure is not perfect, since a permuted trace still contains data about the system being fingerprinted.

Preventing Time Measurements. Countermeasures that reduce, or even disable, the availability of timer APIs can affect our technique, but completely blocking timing measurements from the web is known to be a futile task [188, 189].

3.6.3 Limitations and Insights

Experimental Limitations. The in the wild, crowd-sourced experiments demonstrate that DrawnApart can work successfully in a variety of conditions that are not under the attacker's control. However, our lab experiments only cover a limited set of conditions. Specifically, we only evaluated the impact of temperatures between 26.4 °C and 37 °C, demonstrating no impact on the results. Hence, we cannot preclude the possibility that temperatures outside this range do not affect the results. Similarly, our lab experiments do not control for GPU voltage variations, which could affect our fingerprinting capability. These limitations notwithstanding, the results of the crowd-sourced experiments do provide confidence that DrawnApart is effective in normal operating conditions.

Approach Limitations. We evaluate the effect of device restarts on fingerprinting accuracy by training a model on the Gen 3 devices, and testing the model against traces collected after rebooting the devices. We obtain an overall accuracy of 50.3%. We observe that the accuracy drop is not uniform. That is, some devices maintain stable fingerprints across restarts, whereas the fingerprints of others change significantly each restart. We note that we do not track reboots in our in-the-wild experiments. Hence, these already account for the potential accuracy drop associated with restarts.

We evaluate our technique across ten Chrome versions, from 80.0.3987.116 to 81.0.4044.138. These ten versions consist of two groups: the v80 group which includes six minor versions, and the v81 which includes four minor versions. We train our classifier on the latest v80 version (80.0.3987.163) and test all ten versions. We obtain an accuracy of around 90% on all v80 versions, but significantly lower accuracy, of around 60%, when we test the trained model on v81. We hypothesize some changes in Chrome between v80 and v81 affected the entire WebGL stack. Observing the changelog for the Chromium code repository reveals more than 10,000 commits between the two versions with several hundreds affecting the GPU and the WebGL API [190]. An additional experiment we conducted show that an attacker with a limited trace capture budget can maintain an up-to-date classification model by training a combined model with traces from multiple versions and obtaining a consistently high accuracy of 90±% across all ten versions.

3.6.4 Future Work

In-depth Root Cause Analysis. We shared our work with a committee of WebGL experts in an effort to investigate the root cause of DrawnApart. They acknowledged that the results reported in the paper offer insight on the tracking implications that WebGL can introduce and that our method can highlight differences introduced by the hardware manufacturing process. They propose additional hypotheses for the mechanism through which manufacturing variations enable DrawnApart. Specifically, the two propsals are that:

1. DrawnApart might be uncovering differences in power consumption. A study by von Kistowski et al. [191] noticed differences in power consumption from identical CPUs under the same load but it remains to be seen if and how this could translate to GPUs and WebGL.
2. The effect might be induced by a difference in the response to temperature curves.

Validating either hypothesis requires detailed knowledge of the design and the manufacturing process, which are only available to the manufacturers, and are likely beyond the scope of a typical academic research.

Next-Generation GPU APIs. DrawnApart currently only uses the WebGL API, limiting its speed and accuracy. Upcoming Web-based compute-specific GPU interfaces may allow for far more efficient fingerprinting. There are two compute-specific GPU APIs for web browsers: WebGL 2.0 Compute and WebGPU. WebGL 2.0 Compute was integrated into Chrome but disabled in 2020 [192], and its development has been subsumed by WebGPU [193]. WebGPU is currently under active development, and is not supported in the stable edition of any browser, but preliminary implementations can be found in the canary versions of Firefox, Chrome, and Edge.

These APIs introduce *compute shaders*, a form of computational pipeline that coexists with the existing graphics pipeline. One significant feature offered to compute shaders is the ability to synchronize among different work units, by using atomic functions, message queueing or shared memory. We used this synchronization primitive to prototype a faster fingerprinting technique

for WebGL 2.0 Compute. In our prototype, all workers race to acquire a mutex, and we record the order in which the different work units were granted the mutex. We tested this fingerprinting technique on our Gen 3 corpus, after enabling WebGL 2.0 Compute support in Chrome through a command-line parameter. This compute-based fingerprint delivered a near-perfect classification accuracy of 98%, while taking only 150 milliseconds to run, much faster than the onscreen fingerprint which took a median time of 8 seconds to collect. We believe that a similar method can also be found for the WebGPU API once it becomes generally available. The effects of accelerated compute APIs on user privacy should be considered before they are enabled globally.

3.7 Related work

Web-based Fingerprinting. Eckersley [152] was the first to show that it is possible to fingerprint browsers based on their features and configurations. Mowery *et al.* [143] classified web fingerprinting use as constructive or destructive. Constructive fingerprinting can detect bots [144, 145, 146], or help to protect sign-in processes [147, 148]. Conversely, destructive use can track users and their browsing habits. Many browser attributes are considered parts of a browser fingerprint, including `navigator` and `screen` properties [152, 155], font enumeration [194], audio rendering [150], and the WebGL canvas [86]. These techniques are all unable to tell apart identical devices.

Mobile Fingerprinting. Mobile devices have less hardware and software diversity compared to desktops [195]. However, they possess additional fingerprinting sources such as sensors [196, 197, 198, 199, 200], microphones [201, 202, 203, 204, 205] and cameras [206, 207]. Manufacturing variations can also manifest as differences in the radio frequency (RF) behavior of networked devices [208, 209]. These techniques are tailored to mobile and RF environments, while our technique works in all browsers that support WebGL, without requiring permissions, additional sensors or RF hardware.

Physically Unclonable Functions. The silicon-based physically unclonable function (PUF) concept is based on the idea that, even if a set of several integrated circuits is created through an identical manufacturing process, each circuit is actually slightly different due to normal manufacturing variability. This variability can be used as a unique device fingerprint based on hardware. Examples of silicon PUF sources include logic race conditions [141, 142], Rowhammer behavior [210], and SRAM initialization data [211, 212]. Ruhrmair *et al.* [213] defined a fingerprint as "a small, fixed set of unique analog properties", and explain that the fingerprint should be measured quickly and preferably by an inexpensive device. In this work the GPU is used as a PUF, and our challenge is how to successfully capture the PUF behavior while using the limited APIs available to a web browser.

Responsible Disclosure. We shared a preliminary draft of our work with Intel, ARM, Google, Mozilla and Brave during June-July 2020 and continued sharing our progress with them throughout 2020 and 2021. In response to the disclosure, the Khronos group responsible for the WebGL specification has es-

tablished a technical study group to discuss the disclosure with browser vendors and other stakeholders.

Chapter 4

A Privacy Analysis of Games in Android

Smartphones are in the pockets of 6.3 billion users, which represents more than 80% of the worldwide population [214]. As every single one of these devices has the capacity to play games, the potential market for mobile gaming is huge. In 2021, 2.66 billion mobile gamers [215] spent collectively more than $116 billion USD on mobile games [216], surpassing the revenue of all other gaming sectors combined [217]. As users were faced with a global pandemic, they spent more time at home and on mobile games [218] with a lasting effect and further gains that can already be observed in 2021 [219].

Amidst this booming market, publishers are exploring different ways to monetize their games, as detailed by Tang [220]. More than 95% of games on the Google Play Store are free [221], while the others can be accessed for a relatively small price with most paid games being priced under $10 USD. Games can also earn revenue by showing personalized ads to players or by selling in-app purchases (IAP) (e.g., to unlock levels, buy in-game currency). Inspired by streaming services, some games also offer subscriptions in the form of *battle passes* [222] that offer in-game rewards (e.g., extra content, boosts, skins, new levels). Although each of these methods contribute to generating revenue, as detailed by Unity's Game Report [223], ads and IAPs capture the lion's share of the revenue with IAP slowly starting to surpass advertising revenues in different markets.

Interestingly, at a time when online privacy is at the forefront of discussions regarding the Web, with the development of anti-tracking technologies [224, 225, 226], the upcoming deprecation of third-party cookies [78] and the design of privacy friendly tracking alternatives [227], mobile gaming seems to be have been saved from these discussions. On the one hand, users care about data privacy with 79% willing to spend time and money to protect their data [228]. On the other hand, the pervasiveness of ads and trackers in mobile games seems to be the complete opposite, where users accept opaque data collection and data sharing operations performed by a lot of unknown tracking companies. A survey of US gamers conducted in 2018 revealed that 82% of users preferred free mobile games with ads compared to paid mobile ones without [229]. 74% would also watch an in-game ad if they get an in-app perk in return.

As ads are an integral part of mobile gaming, what implications does it have on users' privacy? What is the true privacy cost of free games? Does paying for a game up front truly ensures better privacy guarantees?

In order to answer all these questions, we analyze in this work 6,751 games, including 396 paid games, and compare the trackers present in them. Section 4.1 describes the related work. Section 4.2 introduces the pipeline that we built to collect our dataset of games. Section 4.3 presents the result of our analysis by considering different dimensions like the number of trackers, the included permissions, the category of a game, its base price and its intended audience. Section 5.5 discusses our findings while Section **??** concludes our paper.

4.1 Related work

In the field of Android security, there is an extensive literature on how to analyze apps, through the use of static analyses to dynamic approaches or even instrumenting firmware or proxying traffic. Some examples include using taint tracking like Flowdroid [230], Taintdroid [231] or AndroidLeaks [232] to capture data leaks, explore how permissions can be circumvented to collect sensitive data [233], or simply look at detecting malware [234]. However, only a handful of studies look at the presence of trackers in mobile applications and if there are differences between free and paid applications.

Tracking in mobile applications Razaghpanah et al. studied the mobile advertising and tracking ecosystem by analysing real-world mobile network traffic [235]. Thanks to an app called Lumen, installed directly on users' devices, they were able to capture where the data from mobile apps was being sent. They also traced back the parent companies behind many different tracking services and found, in particular, that Alphabet was present in over 73% of the 14,599 apps in their dataset. Finally, they discovered that 39% of the tracking services they identified were also present as third-parties in at least one of the Alexa Top 1,000 websites.

Reyes et al. analyzed 5,855 of the most popular free children's apps on Android to see if they were compliant with the Children's Online Privacy Protection Act (COPPA) [236]. They instrumented the APIs that access sensitive resources and used Lumen to detect if data from those APIs was sent over the wire. Their results showed that 19% of tested apps collected identifiers or personally identifiable information that should never have been transmitted.

Kollnig et al. compared the same 12k apps on both Android and iOS to see if there were any differences in terms of privacy [237]. In the end, they found no significant differences between the two platforms despite different architectures and requirements from both app stores. 88% of Android apps had at least one tracking library, while 79% of iOS apps did. In both stores, about 3% of apps had more than ten trackers. Android apps asked for more permissions compared to their iOS counterparts, but this was mainly due to platform differences where some resources on iOS were not gated behind a permission, contrary to Android.

Studying paid applications There are few studies that include paid applications in their datasets, arguably because of the budget required to purchase them. In 2015, Seneviratne et al. collected the top 100 free and paid apps on Android in 4 countries [238]. They found that 60% of the paid apps included at least one third-party tracking library compared to 85% of the free ones.

Moreover, the tracking behaviours of free apps were about the same as for paid ones since they found the same types of trackers in both. In 2019, Han et al. compared 1,505 free Android apps with their paid versions to see if differences could be observed in terms of privacy [239]. About half had an identical set of permissions and third-party libraries between the free and paid version.

More recently, Watanabe et al. performed a large-scale analysis of 2M free apps and 30K paid ones to detect software vulnerabilities [240]. The price of paid apps they studied ranged from $1 USD to $200 USD. By using vulnerability scanners and checking for dead code, they found that 70% of the vulnerabilities in free apps stem from software libraries, compared to 50% for paid ones.

Finally, Ishii et al. looked at the apps present in 13 different Android marketplaces [241]. They observed that some paid apps can be found for free in other marketplaces, with some even having their license verification library removed so that the pirated copy could bypass the control for an existing license.

Our work In this work, we investigate the tracking ecosystem in mobile games as we believe their unique economic models can have an impact on the tracking and advertising libraries that are embedded in them. As the studies discussed above perform measurements on all types of apps without differentiation, we focus here on doing measurements specifically for mobile games. Notably, we want to see if paying for games up front is more privacy friendly than playing free games.

4.2 Dataset

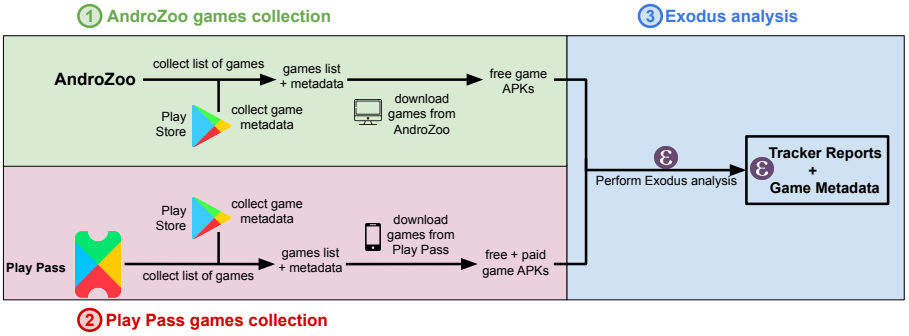
In this section, we detail the dataset that we used to perform our privacy analysis, how we collected it and why we did it that way.

4.2.1 Collecting Android applications

Challenges Collecting mobile games directly from the Play Store is not an easy task. Google provides no list of all the games available on the Play Store. Querying the store through the search bar returns no more than 200 applications. The lists of top applications in different categories are also limited to 200 results. And to make an exhaustive search more difficult, rate limiting is applied on requests coming from the same account and the same IP address. Viennot et al. highlight how complicated it can be to actually collect apps on a large scale with their PlayDrone crawler [242]. They paid participants on Amazon's Mechanical Turk to create legitimate Google accounts to circumvent rate limiting. They also rented Amazon servers to have different IP addresses and queried the store using a 1 million word dictionary to extensively explore the application space.

Collecting free games Because of the limitations imposed by the Play Store and how costly it can be to setup a crawling infrastructure, we relied on the AndroZoo dataset provided by the University of Luxembourg to collect free games [243]. This dataset is regularly updated and, in 2021, contains more than 17 million Android Package Kits (APK) with more than 14 million originating

Figure 4.1: A representation of our pipeline to collect the applications and metadata required for our analysis.



from the Play Store. For this study, we selected all of the free games from 2021 collected in the AndroZoo dataset, which includes games with as few as a dozen active users, up to popular games with millions.

Collecting paid games In order to investigate paid games on the Play Store, we relied on Google's Play Pass [244], a subscription service for games akin to Subscription Video On Demand (SVOD) services like Netflix. By paying a fixed monthly fee, users get access to hundreds of apps and games as part of their subscription and they are all "completely free of ads and in-app purchases" [245]. Through the subscription, paid games can be accessed for free and free games become devoid of in-game ads and in-game purchases. For this study, we collected the 716 games included in Play Pass in November 2021. It should be added that the games downloaded as part of Play Pass are identical to those downloaded by non-Play Pass users. APKs are not built specifically for Play Pass users, everyone downloads the same APK that contains the same code and the same third-party libraries. The only difference is that the Google billing system recognizes if a user has a subscription and provides direct access to games and specific in-app products for no additional charge. This detail is especially important since we want to identify trackers that most users would be subject to and Play Pass gives us access to the proper APKs for our analysis.

4.2.2 Presentation of the hybrid pipeline

Figure 4.1 provides an overview of the pipeline we put in place to collect both free and paid games and analyse their content to identify trackers in them.

Step 1: Collection of free games As detailed in the previous section, we relied on the AndroZoo dataset [243] to collect free games. We downloaded the full list of 17M+ APKs present [246] and identified the 111,035 applications added from the Google Play Store in 2021. This includes apps released in 2021 but also updates to apps released before 2021. Because the metadata in the AndroZoo dataset only includes the app ID, we used a scraper called `Google-play-scraper` [247] to collect additional metadata for each APK directly from the Play Store, such as the app's name, rating and categories. We discarded all apps that were not games and removed the ones that were no

longer available from the store. Our final list includes the APKs and metadata of 6,035 free games available from the Play Store in 2021.

Step 2: Collection of games from Google's Play Pass The biggest difficulty to collect all the games that are part of Play Pass is to get the actual list of what is included in the service. As Google does not provide access to the Play Pass catalogue from a Web browser, we used a Pixel 3 phone and manually added each Play Pass game to the account's wishlist. This way, we could use a Web browser to extract the IDs of all the games that are part of the service directly from the wishlist. Then, for each ID on this list, we used `adb`, the Android Debug Bridge, to direct the Pixel 3 phone to open the corresponding page on the Google Play Store and simulated a tap to proceed to the installation. After the game is downloaded, we extracted it from the phone for further analysis and uninstalled it. At the same time, we used the `Google-play-scraper` to collect the metadata available on the Play Store for each game, just as we did for the free games we collected. In total, we collected from this step 716 APKs, with 396 of them belonging to paid games.

Step 3: Analysis of APKs with Exodus We sent all the APKs we collected in the first two steps to a local instance of Exodus, a privacy auditing platform for Android applications [248]. It statically analyses the content of an APK and returns the list of embedded trackers it has found by identifying specific third-party libraries or URLs associated with tracking companies [249]. It also provides the list of the permissions required by the application. In this study, we adopt the same definition of *tracker* that Exodus uses: "a tracker is a piece of software meant to collect data about you or what you do". As this definition is broad, it means that the trackers reported by Exodus present different levels of privacy intrusions. An ad company that collects the user's geolocation to serve personalized ads is more intrusive than a tracker that only collects bug fixing information when a game crashes. All in all, to paint a better picture of the privacy ecosystem in mobile games, we rely on the 6 different tracker categories that Exodus provides:

- *Advertisements* for trackers whose aim is to serve ads;
- *Analytics* for trackers who collect usage data;
- *Crash reporters* for trackers that report application crashes;
- *Identification* for trackers responsible for determining your digital identity. One example is logging into an app with a Facebook account through Facebook Login;
- *Location* for trackers who determine your geographical location;
- *Profiling* for trackers that are focused on collecting as much information as possible on the user.

Overview of our dataset Table 4.1 provides an overview of the games we collected from our two sources: the free games from the AndroZoo dataset and both free and paid games from Google's Play Pass subscription service. For each game, we have the following data:

Table 4.1: Source of the games present in our dataset.

	Free games	Paid games	Total
AndroZoo	6,035	0	6,035
Play Pass	320	396	716
Total	6,355	396	6,751

Table 4.2: Overview of the presence of trackers in the games of our dataset

	Percentage of games with trackers	Average number of trackers per game	Standard deviation
Free	86.79%	6.11	6.65
Paid	65.31%	1.80	2.38

- the APK with all of the game's files;
- the Exodus report with both the list of embedded trackers and the list of permissions requested by the game;
- the Play Store listing information with the game's name, the age rating, the review scores, the presence of ads, the presence of in-app purchases and the number of installations.

4.3 Analysis

In this section, we aim to understand how various characteristics of a game, such as its economic model, revenue streams, genre, price or target audience influence the presence of trackers. We look at the presence of ads, the initial price of the game, its age rating and its number of users to provide insights into the tracking ecosystem in mobile games.

4.3.1 Impact of the economic model on tracking

Trackers

We find that about 2/3 of paid games have trackers, a 21% reduction compared to free games, and paid games, on average, have fewer trackers than free games. Table 4.2 provides an overview of the presence of trackers in all the games present in our dataset. The majority of free games have between $1 \sim 12$ trackers, while the majority of paid games have between $0 \sim 4$. Looking more precisely at the distribution of trackers in Figure 4.2, 10% of free games have more than 15 embedded trackers, with the highest having 36. For paid games, the top 10% have more than 5 trackers with the highest being 16. These results show as a general trend that paying for games is in general better, from a privacy point of view, but there are examples of paid games with plenty of trackers.

Permissions

As detailed by the official Android documentation [250], permissions are divided into groups with the two main ones being *normal* and *dangerous* (also called

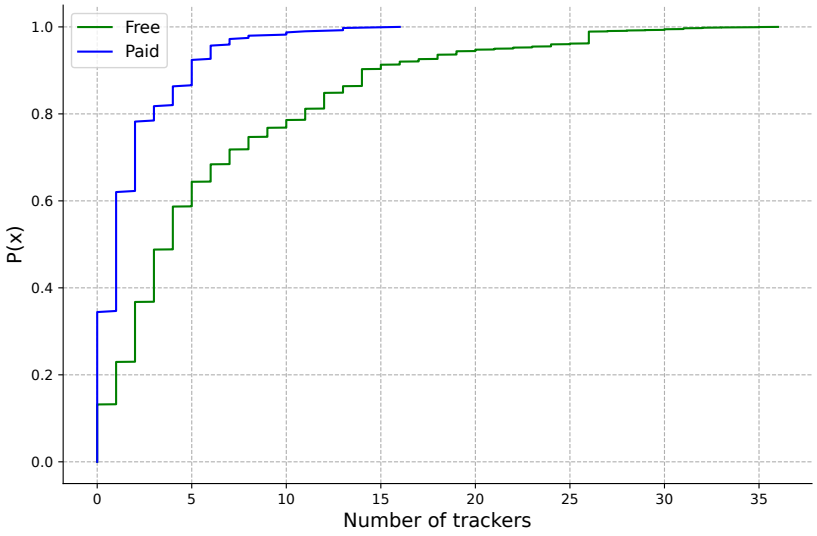


Figure 4.2: Distribution of trackers across free and paid games.

Table 4.3: Top 10 normal and dangerous permissions for free and paid games. Additional information on each permission can be found in the official Android documentation [2].

Normal				Dangerous			
Free		Paid		Free		Paid	
Permission	%	Permission	%	Permission	%	Permission	%
internet	95.32	internet	94.44	write_external_storage	52.22	read_external_storage	31.31
access_network_state	92.43	access_network_state	81.31	read_external_storage	32.66	write_external_storage	31.31
wake_lock	65.69	wake_lock	43.69	read_phone_state	19.18	get_accounts	4.80
access_wifi_state	53.19	access_wifi_state	34.34	access_fine_location	15.49	read_phone_state	2.02
vibrate	37.88	vibrate	21.97	access_coarse_location	15.36	write_settings	1.77
receive_boot_completed	32.43	foreground_service	11.36	write_settings	10.57	access_fine_location	0.76
foreground_service	13.28	change_wifi_multicast_state	7.07	record_audio	6.28	camera	0.51
bluetooth	9.00	receive_boot_completed	5.56	get_accounts	6.09	read_contacts	0.25
change_wifi_state	7.17	bluetooth	2.53	camera	4.16	record_audio	0.25
get_tasks	5.93	modify_audio_settings	2.27	read_contacts	0.29	access_coarse_location	0.25

runtime permissions). A *normal* permission enables access to data and actions that present little risk to the user's privacy, while a *dangerous* one, like the user's location, or contact list, requires explicit consent. We analysed what permissions are used by both paid and free games, with a particular focus on *dangerous* ones:

- Free games: 9.16 permissions on average with 1.52 being dangerous ones.
- Paid games: 6.03 permissions on average with 0.73 being dangerous ones.

Table 4.3 shows the top permissions accessed by most free and paid games. The top 10 normal permissions shows some differences between the two. For example, the `receive_boot_completed` permission shows a 27% difference. According to an official Google forum [251], a change in a dependency in the Google Mobile Ads SDK caused this permission to appear automatically in a lot of applications. Since this SDK is used for Google AdMob, the most popular tracker in our dataset (see Table 4.5), this results in a high use of this permission. We expect this number to even go up as developers update their SDK to the newer version

that includes this dependency change. Other differences exist but the main takeaway is that free apps, on average, request the top 10 normal permissions more often.

For dangerous permissions, access to external storage is at the top for both free and paid games. As mobile games can require extra storage space for textures and assets, it is common for developers to add support for external storage to free up the internal storage. To provide some indication of how large some Android games can be, a popular game called Genshin Impact requires more than 14GB of storage. Location access is high for free games with about 15% of them accessing it, while it is less than 1% for paid games. `read_phone_state` is also high, with 19% of free games asking for this permission. This enables the game to access information like the user's phone number or the current cellular network. Finally, the `write_settings` permission can also prove to be dangerous as the game can modify system settings. In general, we see that paid games ask for less permissions than free games, resulting in less access to sensitive information.

In-game ads and in-app purchases

Table 4.4 splits our dataset into categories based on the presence or absence of in-game ads and in-app purchases (IAPs). First, the presence of either ads or IAPs shows an increase in the overall number of trackers, with an additional increase when both are present. The increase is smaller and more restrained for paid games as can be seen with the smaller averages and standard deviations. Second, there's a strong difference when free games are devoid of ads and IAPs, only 35% of them contain at least one tracker, compared to between 56% and 96% with at least one tracker for other categories. This may indicate that developers looking to monetize their apps are more likely to introduce a third-party tracker. Third, we see that the majority of paid games do not have ads or IAPs, which likely is in accordance with the expectations of consumers who pay up front for a game. However, the majority still do contain trackers. Finally, a curious observation is the presence of advertising trackers in the games that claim they do not contain any ads on the Play Store. This is possibly a limit of the static analysis which we discuss in Section 5.5. Some games might include advertising trackers without using them.

Table 4.4: Overview of the presence of ads and in-app purchases (IAP) in games

Price	Contains Ads	Offers IAP	Number of games	Number with trackers	Avg number of trackers	Stand. Dev. of trackers	Number with advertising trackers	Avg number of advertising trackers
Free	No	No	694	243 (35.0%)	2.00	3.90	156 (22.5%)	0.74
		Yes	512	421 (82.2%)	5.20	4.11	237 (46.3%)	0.92
	Yes	No	3627	3394 (93.5%)	6.15	6.79	3339 (92.1%)	3.12
		Yes	1469	1412 (96.1%)	8.26	7.07	1375 (93.6%)	3.79
Paid	No	No	311	206 (66.2%)	1.68	2.13	77 (24.7%)	0.38
		Yes	58	33 (56.9%)	1.78	2.55	18 (31.0%)	0.50
	Yes	No	8	5 (62.5%)	2.12	2.47	3 (37.5%)	0.37
		Yes	19	15 (78.9%)	3.63	4.39	9 (47.4%)	1.58

Price of games and IAP

Our dataset includes paid games with prices ranging from $0.99 USD to $35.99 USD. The vast majority of games are priced at less than $10 USD. We expected to see a significantly higher number of trackers for cheaper games, for which the price could be justified by higher expected advertising revenues, while more expensive games would need less advertising revenue due to their higher expected sales revenue. However, Figure 4.3 shows that the average number of trackers is not correlated to game prices. Indeed, both free and paid games include monetized content to increase revenue. While we previously noted that IAPs, on average, lead to the presence of a higher number of trackers in games, Figure 4.4 shows that the maximum IAP price does not seem to impact the number of trackers in the game. Similarly, with IAP prices ranging from $1.39 USD to $400 USD, the average number of trackers does not seem to be impacted by the pricing.

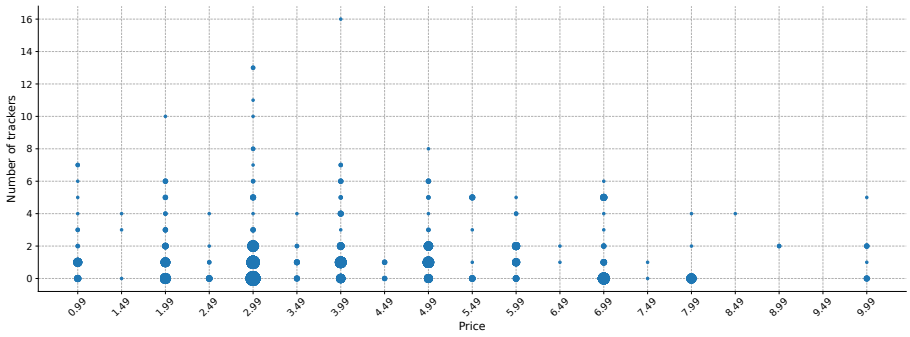


Figure 4.3: Number of trackers per game price.

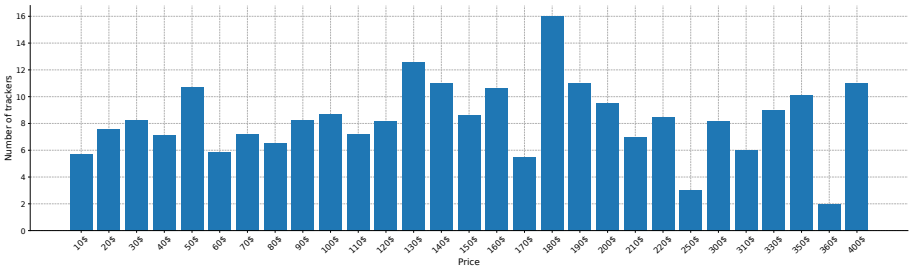


Figure 4.4: Number of trackers per maximum IAP price.

4.3.2 Tracker categories

Figure 4.5 provides an overview of the distribution of trackers across free and paid apps (see Section 4.2.2 for a short description of each category of trackers). The first observation is that advertising trackers are 5 times less present in paid games than in free games. Analytics is the most prominent category of trackers in paid apps compared to advertisements for free games. Then, regarding both profiling and identification trackers, there is a little use of them in free games

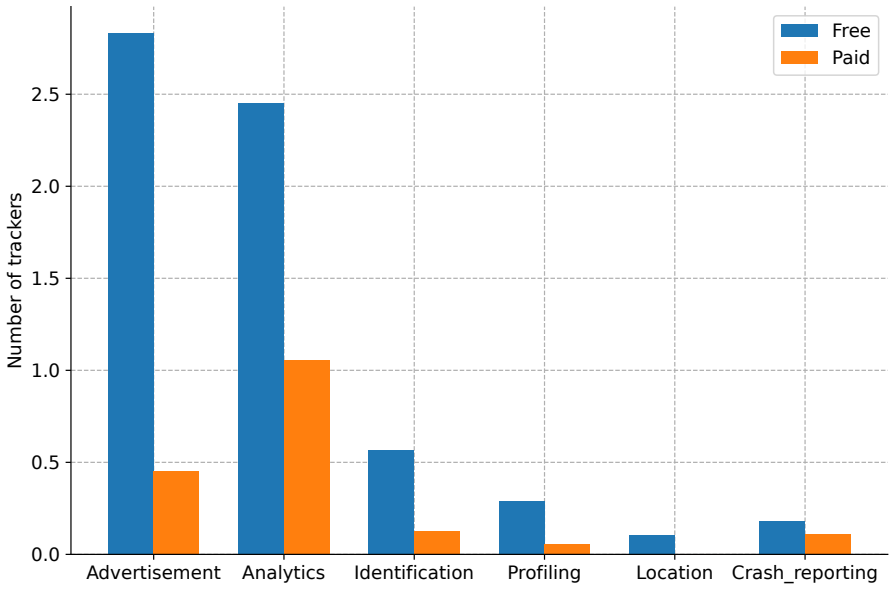


Figure 4.5: Average number of trackers per categories for paid and free games.

and almost non-existent use in paid ones. Finally, Table 4.5 reports on the top 10 most popular trackers across all games. Google has 4 different entries, with each tracker having its own well-defined purpose: Google AdMob serves ads directly in games, Google Firebase Analytics is analytics mainly targeted for developers, Google Analytics 4 is aimed at marketers and includes a "Games report" analysis to provide data on user acquisition, retention, engagement and monetization [252], and Google Tag Manager for managing the tracking tags in a game. Facebook follows closely with two of their own trackers: Facebook Ads and Facebook Login. All in all, the trackers in mobile games are predominately for advertisement and analytics.

4.3.3 Game categories

The Play Store groups games into various categories that the developers choose based on the game's content. Our dataset holds games from 17 categories, with the least represented being the *Casino* category, with a total of 88 games, and the most represented being the *Casual* and *Puzzle* categories, each containing respectively 1168 and 1099 games. Figure 4.6 shows the average number of trackers based on the game categories. *Casual*, *Sports* and *Casino* games are at the top while *Arcade*, *Music* and *Educational* games are at the bottom.

One aspect highlighted by this graph is that the game genre has an impact on the economic model behind a game. The *Casual* category, which is the most represented in our dataset, also includes the highest number of trackers, with an average of over 8 trackers per game. This high number can likely be linked to what *Casual* games offer in terms of gameplay experience. They tend to be played more opportunistically for shorter sessions and they are uninstalled more frequently. They suffer from a lack of player fidelity and higher churn [253], thus

Table 4.5: Top 10 most popular tracker across all games

Tracker name	Categories	Games with this tracker
Google AdMob	Advertisement	4622
Google Firebase Analytics	Analytics	4110
Unity3d Ads	Advertisement	2203
Google Analytics 4	Analytics	1968
Facebook Ads	Advertisement	1758
AppLovin	Analytics, Profiling, Identification, Advertisement	1256
Google Tag Manager	Analytics	1158
IAB Open Measurement	Identification, Advertisement	1115
AdColony	Advertisement	1114
Facebook Login	Identification	1106

introducing maybe the need or the possibility for developers to compensate by serving more ads, more aggressively, to increase revenue quickly. In contrast, game genres where players are more invested long-term, typically include a lower number of trackers. This is the case of the *Action* and *Strategy* categories, which normally capture users for longer sessions. Finally, the *Educational* games are shown to include the least amount of trackers, likely due to the nature of their content and the targeted audience. As detailed in Section 4.3.5, Google imposes strict policies on what can be included in a game targeted for a younger audience. Moreover, to maintain a playful aspect and provide a solid learning environment, game makers may opt to provide less ads to limit disturbances compared to other categories aimed at a more mature audience.

4.3.4 Top 10 games with the most revenue

To dive deeper into how different economic models may impact tracking, we look at the top 10 games with the most revenue on the Play Store in October 2021. Table 4.6 provides the details on these games. Surprisingly, only 3 games out of 10 contain ads, which means they rely mostly on IAPs or subscriptions for revenue. IAPs have a very wide price range, many are under $1 USD with one being as high as $374.99 USD. Regarding trackers, the numbers observed are similar to the average numbers seen in Section 4.3.1 with all of them including analytics and 8 including an identification tracker. Regarding permissions, the numbers vary between the ten games but they all access APIs needed for various features, like access to external storage to download additional assets or to the microphone for multiplayer games that provide audio exchanges during the game. For Pokemon GO, it has the highest number of dangerous permissions but this is also inherent to how the game works. Pokemon GO relies on the player's location to spawn creatures and it uses the device's camera heavily for its augmented reality interface. Both permissions are gated behind explicit prompts because they are considered sensitive for users.

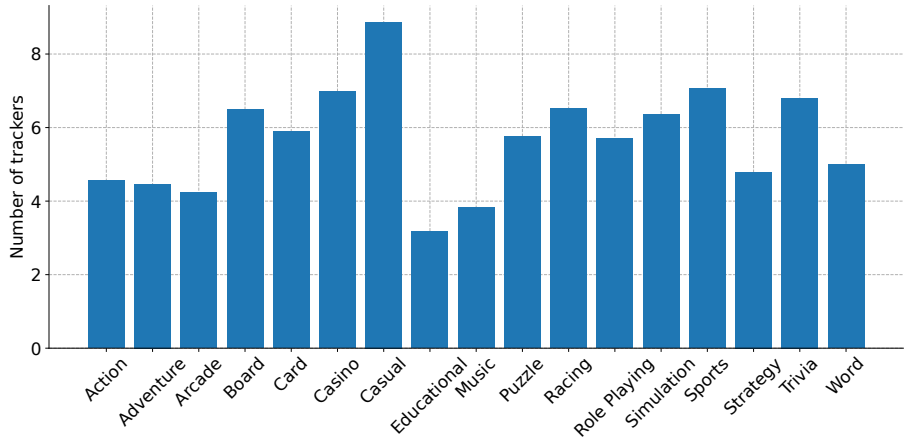


Figure 4.6: Average number of trackers per game genre.

Finally, we can see that the game's category affects the game's source of revenue. For example, there are 3 RPGs in the top 10 that, despite having less than 10M users (compared to Candy Crush Saga's 1B+ users), have no ads and rely entirely on IAPs. This indicates that the type of mobile game can attract different player bases with different spending habits.

In the end, the top 10 provides a glimpse at how popularity and genre can impact the economic model of a game. While less popular games may rely on ads and trackers to bring in revenue, more popular games can also sustain themselves exclusively on IAPs, even in cases when they are otherwise free games.

4.3.5 Google's "Teacher Approved" games

With children increasingly relying on technology for both education and entertainment, Google has recently unveiled a new gaming section labelled *Teacher Approved*. Apps that are designed for children must participate in a larger *Designed for Families* program [254], which opens their eligibility to be rated for the *Teacher Approved* program, without being guaranteed of inclusion. The program includes stricter policies that apps must follow in order to obtain the certification. These policies are verified by a panel of U.S based specialists, which includes teachers, and requires that the app's content and functionality be accessible and appropriate for children. Another major point of the *Designed For Families* program is that it imposes limits on advertising and tracking for children. Developers must follow the *Family Policy* when targetting children with ads, they can either take on these additional responsibilities when using in-house advertising, or they may use one of the self-certified ad SDKs [255]. Naturally, given these restrictions, approved apps are expected to contain less tracking and advertising libraries.

Our dataset includes 181 games with the *Teacher Approved* label, of which 110 are available for free. Our analysis shows that 75% of those games include at least one tracker, with almost 47% including at least one advertising tracker. As these results may look surprising, they are not unexpected as Google does

not restrict approved apps from including trackers, but rather requires them to abide to stricter tracking and advertising practices, including using certified ads [256].

Table 4.6: Information on the top 10 games with the most revenue on the Google Play Store in October 2021

Revenue rank	Game category	Contains ads	Range of IAP	Number of installs	Number of trackers	Number of permissions (dangerous)
Garena Free Fire	Action	No	$0.99 USD - $109.99 USD	1,000,000,000+	7	28 (6)
Candy Crush Saga	Casual	Yes	$0.99 USD - $149.99 USD	1,000,000,000+	7	11 (0)
Coin Master	Casual	No	$0.99 USD - $374.99 USD	100,000,000+	8	13 (1)
Odin: Valhalla rising	RPG	No	$4.99 USD - $89.99 USD	1,000,000+	3	16 (1)
PUBG Mobile	Action	Yes	$0.99 USD - $199.99 USD	500,000,000+	12	23 (4)
Pokémon GO	Adventure	No	$0.99 USD - $99.99 USD	100,000,000+	9	26 (7)
Roblox	Adventure	No	$0.49 USD - $199.99 USD	500,000,000+	3	13 (3)
Genshin Impact	RPG	No	$0.99 USD - $99.99 USD	10,000,000+	6	14 (3)
Gardenscapes	Casual	Yes	$0.49 USD - $99.99 USD	100,000,000+	11	13 (2)
Fate/Grand Order	RPG	No	$0.99 USD - $79.99 USD	1,000,000+	3	12 (1)

Teacher Approved apps still include a significantly lower number of trackers, with, on average, 2.12 trackers per game compared to 6.11 trackers for free games. We also note that the top three trackers used in *Teacher Approved* games make use of Google's APIs, namely, Google Firebase Analytics, followed by Google AdMob and Google Analytics. We notice however that other advertiser's APIs are also served. Although Google requires that only self-certified ad SDKs are authorized to serve ads to children in certified apps [255], we do note that 29 *Teacher Approved* games include trackers flagged by Exodus as Advertisement trackers that are not present in the list of Google's self-certified ad SDKs.

Figure 4.7 provides an overview of the average number of trackers split by different categories. It can be seen that even though profiling goes against the guidelines, trackers within this category can still be found in *Teacher Approved* games. We can however notice that the vast majority of trackers belong to the Analytics category, which follows with our previous observations. Overall, *Teacher Approved* games do seem to provide, from a privacy point-of-view, a better experience, with less ad and tracker frameworks.

4.4 Discussion

4.4.1 Summary of findings and privacy implications

In this study, we saw that a variety of economic models are being used to monetize mobile games. Some games favour ads and in-app purchases for revenue, while others rely on the more classical approach of being purchased for an up front fee. Our analysis reveals that paying for a mobile game leads to, on average, a smaller number of trackers and little to no ads. Analytics are the most prominent form of tracking in paid games, while advertisement is very prominent in free ones. Different genres also have an effect on the number of trackers as the ways they engage users and encourage spending can vary greatly, as can be seen by the revenue models between a match-3 puzzle game and a fantasy RPG.

Yet, one of the most important results from our study is how developed the tracking industry is in mobile gaming. Out of 6,751 games, more than 85% had at at least one tracker embedded in it. In terms of privacy, this number paints a bleak picture as a lot of data is being collected on what the players do and how they play. Even though not all collected data pertains to the player's exact identity, a lot of it is still linked to a virtual identity and likely passed around for analyses and monetization between many companies. This shows that users are under constant scrutiny when using their smartphones for gaming, as a single tracker has the capacity to record anything from the smallest gestures up to getting the user's list of contacts.

Discussing monetization in mobile games would not be complete without mentioning the current state of the ad industry. As privacy is being put at the forefront of discussions about users' digital well-being, alternatives are being designed to protect users from invasive tracking techniques and it is leading to strong changes in the mobile ad ecosystem. A first example is the *AppTracking-Transparency framework* by Apple [257]. When launching a new app, the user can chose not to share their device's advertising identifier, which means trackers

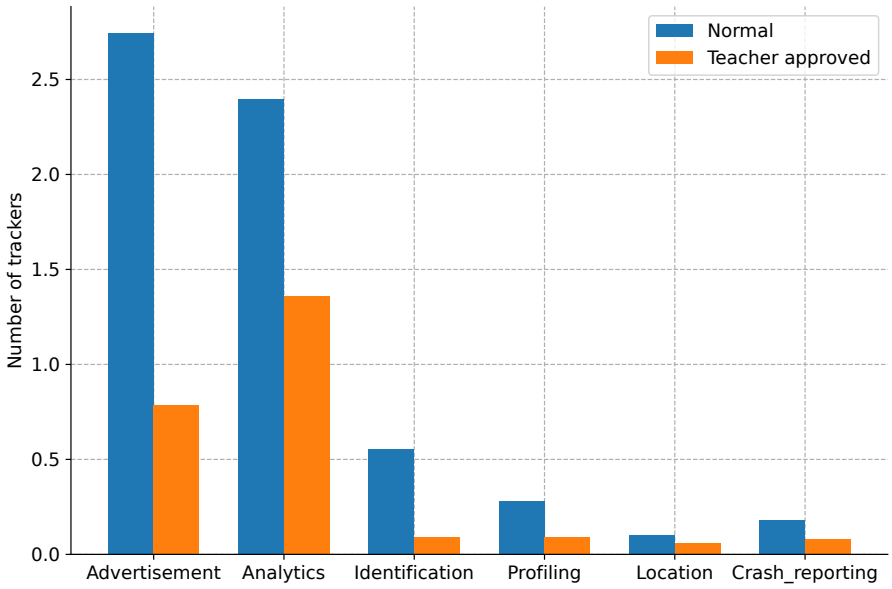


Figure 4.7: Average number of trackers per categories for Teacher Approved and regular games.

cannot link the activity of the user across different apps on the same device. While this does not prevent the collection of information, it limits the creation of very large user profiles based on the data from dozens of apps. Another example leading to changes is the return of contextual advertising [258]. Instead of personalizing an ad using all the information collected on the user, the ad will be based on the content of the page that the user is seeing. The intrusiveness of this technique on user's privacy is minimal as companies do not need to build and maintain user profiles based on behaviour, purchasing habits or other factors. In the end, it remains to be seen the impact that these changes will have on trackers in mobile apps and if it will indeed lessen the heavy scrutiny that users are subject to, often without their knowledge.

4.4.2 Limitations and future work

A first avenue for future research is to use dynamic analysis to go deeper into the analysis of tracking in mobile games. While static analysis reveals the presence of specific trackers, we do not capture how they are actually used in games and if they are triggered at all. Using a tool, such as Lumen [235, 236], combined with complex scenarios to exercise the game and explore as many options as possible, would likely help us identify the information being sent, how sensitive it is, and what the final destination is.

A second avenue is to look at games provided by marketplaces other than Google's Play Store, and platforms other than Android. As each marketplace has its own requirements when submitting an app, tracker analysis would reveal if players from different marketplaces are subject to less or more tracking than those who rely on the more popular Play Store.

Finally, a third avenue is to integrate the future evolution of the Android platform in a tracking analysis. Google has just announced that in February 2022, they will show a *Data Safety Section* on apps in the Play Store to indicate what user data each app collects and shares [259]. Integrating this data into our study could help refine its findings. Another evolution is the new Android App Bundle (AAB) format for Android applications [260]. Designed to be more flexible than the traditional APKs, apps delivered with the AAB format will be optimized for the user's device based on its configuration and language. What remains to be seen is how tracking companies will utilize this mechanism, for example, to deliver different trackers based on the device used by the user.

Chapter 5

Environmental Impact of Online Ads

5.1 Introduction

As of July 2023, there were over 5.19 billion Internet users in the world [261]. This ever-increasing online population demands new services and new content to be delivered at the highest quality and lowest latencies possible. In 2018, it was estimated that over 2.5 quintillion bytes of data were being created daily [262]. And interestingly, much of the Internet is free for users to access, financed indirectly through advertising. To increase revenue, the advertising industry has evolved from simple banners to a complex ecosystem that relies on tracking and profiling users [263, 264].

However, the prevalence and invasiveness of ads increasingly questions their sustainability. Indeed, during much of the same period that online advertising developed, the consequences of climate change have become better understood and more visible. According to the United States Environmental Protection Agency global *greenhouse gas* (GHG) emissions have increased by 43% between 1990 and 2015 [265]. *Data centers* (DC) host the majority of the content being accessed online and account for over 1% of global electricity demand [266]. As such, many previous works have focused on assessing and optimizing DC carbon footprint [267, 268, 269, 270].

Little has been done to assess the carbon footprint of online advertising. Taylor *et al.* [13] estimated in 2008 that every million ad impressions produces 676 $kgCO_2e$. In 2018 Pärssinen *et al.* [138] proposed a layered model and reported that digital advertising is responsible for 60 $MtCO_2e$. To date, these two studies are the most extensive estimates of the carbon footprint of the advertising ecosystem. However, they are based on estimates and simulations and, to the best of our knowledge, no real-world measurements have assessed their conclusions.

In *Ad-Carbon* we report on a real-world measurement of the carbon impact of the advertising ecosystem. Our contributions are:

1. An approach that leverages end-to-end measurements to estimate the carbon footprint of advertisements. Specifically, we propose a model that quantifies the carbon footprint of the client device through fine-grain per-event profiling and energy consumption monitoring, the network through routing data and latency measurements, and the server-side through response time measurements, allowing us to estimate the carbon emissions of individual requests. Our model factors in the electricity mix of the

country through which the request is routed. We assess the stability of our approach through four crawls performed between June and October 2023.

2. We provide an end-to-end estimate of the carbon footprint of different categories of websites by visiting over 31,300 webpages from over 10,500 domains of the Tranco list [271]. Our crawls are done both with and without ad-blocking, as well as with and without consenting to cookie banners. We find that advertisements are responsible for an increase of 144% of the carbon footprint of our crawls. We identify *scripts* to be responsible for the majority of CO_2eemissions of our crawls, with 41.36 $kgCO_2e$ emitted.
3. Through *prebid.js*, we leverage client-side header bidding to estimate the carbon footprint of ad auctions and show they generate an additional 0.16 gCO_2e, on average, per visit.
4. We instrument *cookie banners* to show that the acceptance of cookies (and trackers) produces 90% more CO_2e.

5.2 Background & Motivations

Advertisements generated over $209B in 2022 [272], an increase of 10.8% over 2021, which increased again by 36% over 2020. Advertising is present in every aspect of the Web. From search engines to video streaming platforms, resource-consuming ads are pervasive and constantly require new infrastructure to keep up with the demand. In 2015, Pujol *et al.* [273] estimated that over 18% of HTTP requests performed by Web pages were related to advertising. These results are confirmed by Gui *et al.* [274], whose work shows that *"apps with ads consume, on average: 48% more CPU time, 16% more energy, and 79% more network data."* These studies help to understand the impact of advertising on the end user's device, but do not account for the network and backend costs of advertising.

Much of online advertising relies on profiling and tracking [263, 264]. Cookies, cookie sharing [69, 72], trackers [65], as well as browser fingerprinting [6, 8, 15] directly increase energy consumption and carbon emissions on the user's device, on servers and on networks. Interestingly, a study in 2020 showed that uBlock Origin[1], a popular ad-blocker [275], reduces page load times by over 28% and could save users over 100h of electricity each year, a significant reduction to their carbon footprint [276]. This is a lower bound as they do not account for network or server-side resources.

5.2.1 Ad bidding

Real-Time Bidding (RTB) [277], introduced in the late 2000s, allows advertisers to purchase individual ad impressions on publishers' websites through an *ad exchange platform* using an auction-based protocol. Advertisers choose to bid based on the user's profile. Bids are sequentially collected from *Demand-Side Platforms* (DSPs), in a waterfall process, and the winning bid is displayed in the publisher's slot.

[1] `https://ublockorigin.com/`

This sequential process makes it easy to miss higher bids. *Header Bidding* (HB) [106] flattens the process and reaches out to all registered ad exchanges or *Supply-Side Platforms* (SSPs) at once. Header bidding comes in two main forms:

- *Client-side header bidding* runs in the client's browser. The publisher defines slots and the exchanges for each auction. The client's browser initiates the auction through requests to different exchanges. The exchanges then might decide to bid. The exchanges respond with an object containing the bid value, if any, and the ad's content. Once all bids are collected or the timeout is reached, the client identifies the winner and notifies the result. While client-side header bidding provides many benefits to the publisher, including cookie syncing and re-targeting, it remains limited by the client device's network performance.
- *Server-side header bidding* is newer and solves the resource issues of client-side header bidding by running in a server. Both processes are similar but latency is improved and the number of exchanges is not limited by the client's device, potentially increasing the top bid. As server-side header bidding does not happen in the client's browser, it does not include client-related cookies and, therefore, a less precise profile of the user is sent to the advertisers.

Pachilakis *et al.* [107] show that over 14% of the Top *35,000* Alexa websites implemented *client-side header bidding*, and that *header bidding* significantly increases the time to display an ad. Our study relies on client-side header bidding to estimate the carbon footprint of the auction process by analyzing events through *prebid.js*,[2] an open-source bidding platform. Based on *prebid.js*, Pachilakis *et al.* [107] show that despite reaching more ad exchanges, publishers collaborate with fewer, more efficient exchanges. Cook *et al.* [278] leverage *prebid.js* to build a model to infer data-sharing between trackers and advertisers that are missed by cookie syncing.

5.2.2 Impact of advertisement

It is estimated that the Internet user over 10% of global electricity and is rising [132, 133]. This is a significant carbon footprint. As the advertising industry has evolved into a core component of the Web, we provide estimates of the carbon footprint of online advertising. Few studies have taken a look at the energy consumption and carbon footprint of online advertising [13, 138]. These publications lack detail, mostly depicting the industry's carbon footprint from a high level. The reasons for the lack of literature are multiple:

- Researchers rarely have access to the infrastructure used by advertisers and cannot measure their activities. Therefore, we rely on estimates with high degrees of uncertainty.
- Due to the increasingly complex processes of the advertising ecosystem and the reliance on real-time bidding, ads often involve many network requests, each contributing to the ad's carbon footprint.

[2] `https://prebid.org/`

Gonzalez *et al.* recently released CarbonTag [139], a framework that attempts to measure the carbon emissions of individual advertising units from CodeCarbon [279] measurements. Their approach relies on a trained ML model that approximates the emissions of the rendering process of a displayed advertisement unit. CarbonTag achieves high accuracy but remains limited to the client's device. Nonetheless, the authors acknowledge the need for a framework that covers the end-to-end ad process. To the best of our knowledge, CarbonTag is the closest work to our study to date.

Our work is, therefore, motivated by the need to estimate the end-to-end carbon emissions of the online advertising industry. We show that through the use of SmartWatts [14] on end devices, large coordinated crawls to access Web pages and detect advertisements, and the instrumentation of client-side header bidding, we can accurately measure the carbon footprint on the client device and approximate the network and data center costs through both an extensive study of the state-of-the-art and collected measurements.

5.3 Measurement Methodology

We propose a model that estimates the end-to-end carbon footprint of online advertising. We rely on Web crawls and detailed client-side instrumentation to identify ad-based network requests, while monitoring client-side energy consumption. We estimate the server-side energy consumption of ad requests by stress-testing a mock-up website running on standard rack servers and observing power consumption on different loads. Finally, we complete our model with network measurements to estimate the network costs of ad requests. In particular, we note that our approach gains new insights thanks to client-side header bidding, which makes the bidding process more transparent. In practice, we perform two crawls:

1. In the *Regular* crawling session we crawl over 10,000 of the top domains of the Tranco list, twice, once with and once without blocking ads. This process is repeated four times to assess the stability of our measurements. We compare the crawls with and without ad blocking to calculate the carbon footprint of advertising, and we use instrumentation to estimate the cost of *header bidding*.
2. The *Cookies* crawling session focuses on the carbon footprint of consenting to targeted advertising and trackers through cookie banners. We crawl a selection of domains from the Tranco list, twice. Once accepting the banner (which generates additional activity like ads and trackers), and another without. We compare them to estimate the carbon footprint of accepting banners.

Seeding the profile Previous research shows that advertisers are more prone to display ads to users with an established browsing history and profile [278, 280, 281, 282]. To build a profile for our crawls, we sequentially visit three pages from each random website from the top 20,000 most popular domains from the Tranco list [271] (version *25L99*). We start the profile from a clean state—*i.e.*, no cookies, empty cache. Over 24 hours, we visit each website, scroll the page and open a URL of the same domain in a new tab. We scroll again and open

a third page for each domain. The seeded profile is used for the main crawling sessions, which are parallelized. Our approach can be summarized as:

1. Seed a profile for 24 hours to signal advertisers, and crawl over $10,000$ domains from the Tranco list (section 5.3.1).
2. Extract client-side power and network measurements to compute the carbon footprint for each requested ad and estimate its server and network energy (section 5.3.3).
3. Perform an additional crawl, with and without ad blocking, to estimate the carbon footprint of cookie banners (section 5.3.1).

Figure 5.1 gives an overview of the infrastructure.[3] For all our crawls, we use the following three devices:

- One Intel NUC10i7FNH with a 12-core Intel i7-10710 CPU and 32 GB of RAM, running on a minimal version of Ubuntu 22.04.2 LTS (kernel 5.15.0-75-generic)
- One Intel NUC9i7QNB with a 12-core Intel i7-9750H CPU and 32 GB of RAM, running on the same minimal version of Ubuntu (kernel version 5.15.0-73-generic)
- The main node, which instruments the crawls and hosts the database, runs on a server with two 24-core Intel Xeon Gold 5118 CPUs and 188 GB of RAM.

First, the main node curates a set of websites and instructs devices to crawl them. Each device starts the power consumption probes, using SmartWatts [14], and launches parallel browser instances to crawl the websites. We use unique IP addresses for each device to reduce the risk of being flagged as a bot. Finally, we collect the resulting crawl data and compute the CO_2eq emissions associated with each request.

[3]The research artifact accompanying this work can be found at `https://github.com/naifmeh/adcarbon`

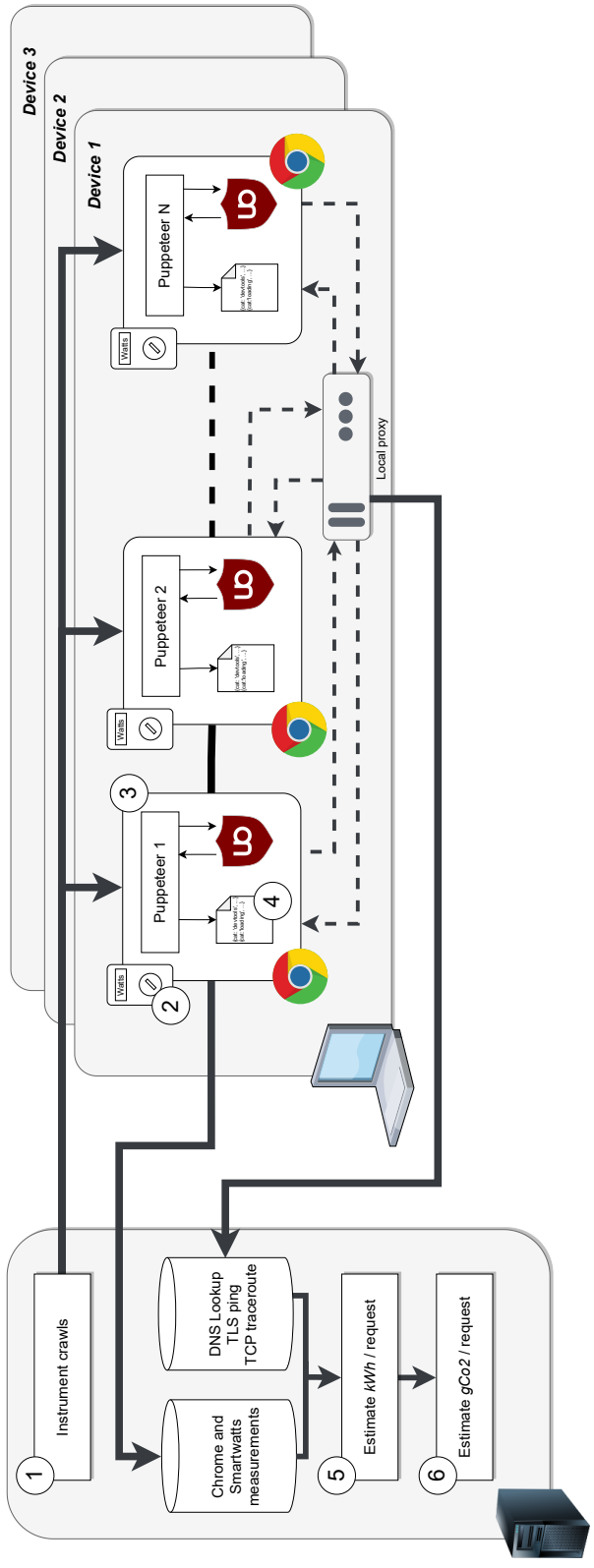


Figure 5.1: Infrastructure. Each device has a unique IP address. ① The main server instructs the devices to crawl curated lists. ② For each browser instance, a SmartWatts probe is initialized. ③ The crawls start and requests are processed. ④ Chrome traces are collected at the end of each crawl, for each page visited. ⑤ The crawling data and energy consumption are collected, and ⑥ the CO_2emodel is applied to estimate the carbon footprint of each request

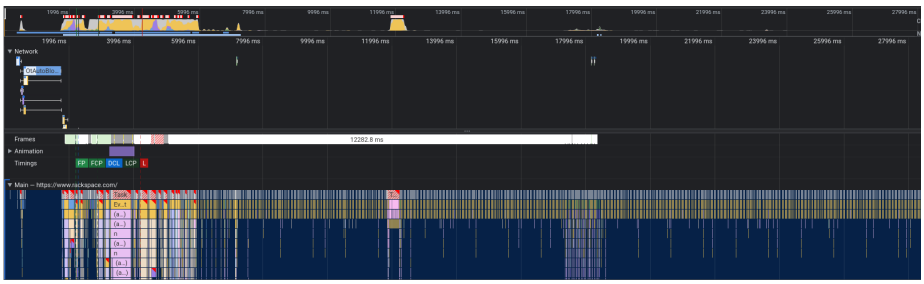


Figure 5.2: Visual sample of a tracing file in Google Chrome

5.3.1 Web crawling protocol

We use PUPPETEER[4] and instrument *Google Chrome* (version *114.0.5735.106*) for all our crawls. Our choice is motivated by Chrome's market share of over 64%.[5] Instrumenting browsers other than Chrome is out of the scope of our study. Following best practices [283, 284], we use the *Puppeteer extra stealth* add-on.[6] which provides tools to make PUPPETEER look like a genuine browser and reduce the risk of being blocked. We use Chrome's Tracing API to record a trace file with fine-grained data about each event on the page, as well as its execution stack. This allows us to measure how long it takes to render an image and which process was used. The trace file contains information about interpreting and compiling scripts, and all function calls and their stack traces. We correlate this information with our SMARTWATTS measurements for a fine-grained estimate of the impact of each request on the client's device. A sample tracing file can be observed in Figure 5.2.

We use browser profiles to save the browser's state. For the *Regular* crawling session, each crawl uses the same seeded profile. Because profiles can not be shared by browser instances, we clone the profile on each device n times, with n being the number of parallel instances (in practice, we use 8). For the *Cookies* crawling session, we use clean profiles. n profiles are used for the execution of the first scenario in which the crawler does not interact with the cookie banner, and another n profiles are used in the scenario that accepts the cookie banner.

Power consumption We use SMARTWATTS [14], a *state-of-the-art* self-calibrating software-based power monitoring system to measure the energy consumption of every browser instance. SMARTWATTS supports per-process measurements, meaning we can run parallel browser instances with minimal impact on the measurements. We isolate each browser in a new *control group* (cgroup).[7] At the end of each crawl, we retrieve the power measurements for the cgroup the browser was affected to.

[4] https://developer.chrome.com/docs/puppeteer/

[5] https://gs.statcounter.com/browser-market-share/desktop/worldwide

[6] https://github.com/berstend/puppeteer-extra/tree/master/packages/puppeteer-extra-plugin-stealth

[7] https://www.kernel.org/doc/html/latest/admin-guide/cgroup-v1/cgroups.html

Proxy We implement a *Man-In-The-Middle* (MITM) proxy in Rust to optimize for performance. Our proxy parses request headers and forwards requests from crawled webpages to the Redis queues. The proxy also acts as a network ad-blocker, intercepting requests containing the *block* attribute that was added by our modified ad-blocker (described in the next paragraph). Additionally, for each request, the proxy logs information to calculate the carbon footprint, such as the total number of bytes transmitted, the duration of the request in nanoseconds, and the type of resource.

Cookies & Ads We use a modified version of *uBlock Origin* [285] to identify and tag advertisement requests. To achieve this, we disable uBlock's ad-blocking mechanism and instead modify uBlock to tag the request's header attributes if it's an ad. This header is later used by the proxy to either allow or block the request, depending on whether we want the webpage version with or without ads. Furthermore, the *Easylist Cookie List* filter list contains known selectors for various cookie banners. We use these to detect banner prompts whenever possible. We use a two-step pipeline to interact with the banners: first, for each selector matched on a given page, we click on occurrences or variations of consent sentences for all 24 official languages of the European Union. We focus on European languages because they are more likely to implement cookie banners due to the GDPR.[8] Second, if no selector is found, we look for exact occurrences of a set of consent wordings that we manually collected from two of the top 10 websites of each EU country. This methodology is motivated by previous works on consent banners [286].

Network For each page we visit, we instrument the crawler to collect information on every request and response. Requests are processed in the proxy and added to three Redis[9] queues:

1. The first queue resolves the domain's IPv4 and uses Telize and Maxmind's City GeoIP2 database[10] to obtain the country and region where the request is sent. The IP is then forwarded to the two next queues.
2. The second queue uses Hping3 to perform a simple *TCP SYN* ping and measure the time it took to reach the IP. This value is later used to differentiate the server's processing time from the network latency.
3. The third and final queue uses the *traceroute* command to issue TCP probes (as opposed to ICMP probes) that aim to trace the different hops a packet has to perform before reaching its destination. For each hop we identify, the IP address is forwarded to the first queue to identify its origin. This IP does not continue to the second and third queues since it's a router and not a server.

Header bidding For each page we visit, we collect bidding data through the header bidding mechanism. Following previous work [287, 280], we collect bidding data on domains that use the *prebid.js* library. Before closing each page, we leverage *prebid.js* to get the page's ad units, all the bids that were performed, their respective bidders, and the winning bids.

[8] `https://gdpr-info.eu/`
[9] `https://redis.io/`
[10] `https://www.maxmind.com/en/geoip2-city`

Browsing scenario For the two crawling scenarios, we set a timeout value of 60 seconds upon the firing of the networkidle2 event. We wait 5 seconds before interacting with the page. We visit over 10,000 domains from the top 20,000 domains of the TRANCO list. We crawl 3 pages for each domain. All pages must succeed in loading to consider the crawl successful.

The two crawling sessions differ slightly in their course:

- For the *Regular* crawling session, we visit 3 pages for each domain without blocking advertisement requests. The sequence is stored. We then enable ad-blocking, through the proxy, and reproduce the visit. When possible, we click cookie banners to accept all cookies otherwise some websites do not activate the ad-bidding process.
- The *Cookies* session is similar with the exception that the ad-blocking tool is never enabled. We perform the same visit scenario twice, once without interacting with the cookie banner and the second accepting all cookies where possible.

5.3.2 System boundaries

The Internet's complexity makes precise measurements complicated when control of each part of the pipeline is not possible. Therefore, the literature defines system boundaries [288, 138], which typically include a data center and a simplified network consisting of a few nodes (routers), but exclude the terminal device. We consider the following three main subsystems (which can be visualized graphically in Figure 5.3): 1. the client device used to access and process Web pages, 2. the network, from the user's router to the data center's entry point, and 3. the data center that serves the user's requests and resources.

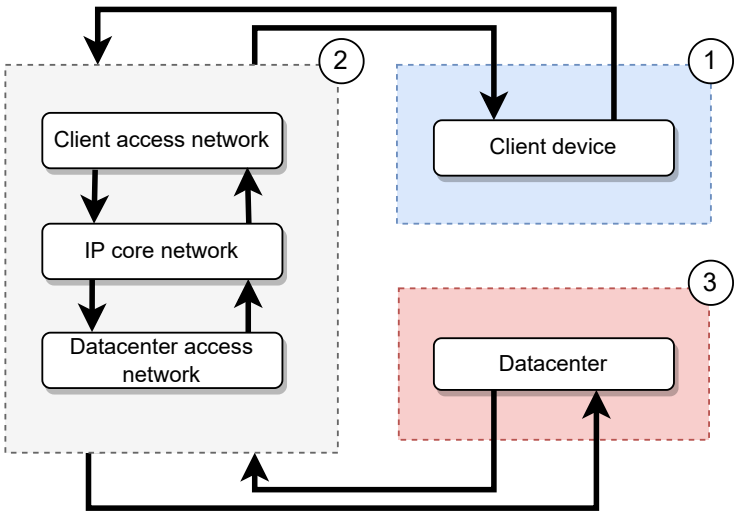


Figure 5.3: System boundaries underlying our study. We consider three main subsystems consisting of ① the client device, ② a simplified network, and ③ the data center. The arrows represent the data flow.

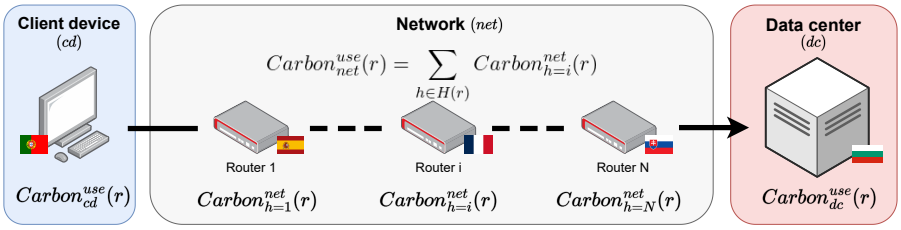


Figure 5.4: High-level representation of the carbon footprint computation method for each request. In this example, the request is emitted from Portugal and transits multiple European countries before reaching its destination in Hungary.

5.3.3 Carbon footprint model

Visiting a given website generates a set of requests R to remote servers (generally in data centers). Therefore, the carbon emissions of a website visit of duration d is given by the sum of the carbon emissions of each emitted network request r:

$$Carbon^{use}_{visit}(d) = \sum_{r \in R(d)} Carbon^{use}_{visit}(r) \tag{5.1}$$

Figure 5.4 shows a high-level depiction of the flow of a network request. As such, the in-use carbon footprint of a single request r can be modeled as the sum of the *client device* (cd), network (net), and data center (dc) carbon emissions:

$$Carbon^{use}_{visit}(r) = Carbon^{use}_{cd}(r) + Carbon^{use}_{net}(r) + Carbon^{use}_{dc}(r) \tag{5.2}$$

The client device's carbon emission $Carbon^{use}_{cd}(r)$ for each request r can be estimated by multiplying the measured power usage, defined by $\overline{P}_{cd}$ (obtained from the SmartWatts software probe), the response's latency (time it took to parse and interpret the response) $Ltc_{client}(r)$, and the client country's energy mix $E_{mix}(Src(r))$ at the time of the request:

$$Carbon^{use}_{cd}(r) = \overline{P}_{cd} \times Ltc_{client}(r) \times E_{mix}(Src(r)) \tag{5.3}$$

SmartWatts estimates the power consumption of a process and is not fine-grain enough to estimate the energy consumed by a single request. To overcome this challenge, we use Chrome's built-in Trace Event Profiling Tool, or more simply, *tracing API*. The tracing API provides a very fine-grain report of the execution sequence of a visited page. It outlines the different processes, threads and the tasks they execute at the microsecond granularity. For instance, as a script is requested, it is possible to extract information about the script's parsing time, the compilation time, and its complete stack trace, which includes every function called and their duration. For each data point, it is possible to obtain the associated thread and process. Thus, for each request r, we estimate $\overline{P}cd$ and $Ltc_{client}(r)$ by identifying all the related events from the tracing file, their duration, and their *process ID* (PID). We evenly distribute the measured consumption of SmartWatts for the event durations among all the processes that are active simultaneously.

The in-use carbon emission of the network involved in a request r can be estimated as the sum of the carbon emissions of all hops H in a network route (obtained via a TCP traceroute):

$$Carbon^{use}_{net}(r) = \sum_{h \in H(r)} GBytes(r) \times \overline{P}_{router} \times E_{mix}(Hop(r)) \tag{5.4}$$

where the energy mix is assumed to be the country's (obtained through MaxMind GeoCity2 database) energy mix of each IP that corresponds to each hop and $GBytes(r)$ is the number of bytes that transit through each router. $\overline{P}_{router}$ is defined by Schien *et al.* 's mean consumed power per gigabyte of transmitted data for metro routers [289]. This constant is expressed in kWh/GB and is equal to $3.56 \cdot 10^{-5} kWh/GB$. Since the first N hops will always transit through our local Renater network, we use the constant that was estimated by Ficher *et al.* in their Renater carbon footprint report [290]. This constant is set to $3.68 \cdot 10^{-4}\ kWh/GB$.

We're unable to report a more accurate value of $Carbon^{use}_{dc}(r)$ that would suit each and every data center due to the opacity of the Internet's backend. Therefore, we estimate the data center's carbon footprint for a request r as:

$$Carbon^{use}_{dc}(r) = \overline{P}_{dc} \times Ltc_{dc}(r) \times E_{mix}(Dst(r)) \tag{5.5}$$

where $\overline{P}_{dc}$ is the average power consumption of the server and $Ltc_{dc}(r)$ is the time spent on the server. We estimate data center latency, $Ltc_{dc}(r)$, by subtracting the latency of a *TCP SYN ping* to the server and the total latency of the request. We chose to use TCP SYN ping instead of a simple ICMP ping because it provides a more accurate representation of the time a packet transits the network, as ICMP pings are often given low priority or discarded. Finally, $E_{mix}(Dest(r))$ captures the energy mix consumed by the remote server, which can be obtained by determining the location of the server (*e.g.*, via the IP address) and the energy mix of the associated region.

As we do not have control over the chosen server, we estimate $\overline{P}_{dc}$ through a series of measurements on a remote server under our control consisting of 2 *Intel Xeon E5-2630 v4* CPUs, 256 GB of RAM and running a minimal version of Debian 9 to minimize the impact of other processes. We implemented a control website using multiple web application backends and selected NodeJS as our reference since it exhibits the median performance. Appendix B.1 compares the behavior of other backends. The average power of the web server evolves based on the number of concurrent clients. After putting the server through a warm-up phase, for each fixed number of concurrent clients, we stress the server with millions of requests during 20 seconds. Then, we approximate P_{dc} as:

$$\overline{P}_{dc} = \frac{\overline{P}_{measured}}{N_{clients}} \tag{5.6}$$

where $\overline{P}_{measured}$ is the average power measured during the stress test and $N_{clients}$ is the number of concurrent clients. Moreover, our measurements show that beyond 32 concurrent clients, the web server starts to be saturated, and the power consumption remains constant above this point. Hence, we set $N_{clients} = 32$ for

our measurements. Our experiments led us to set $\overline{P}_{measured}$ to 103 $watts$. Additionally, we multiply the result of the previous formula by an average *Power Usage Effectiveness* (PUE) of 1.12, as estimated by Liu *et al.* [291].

For each contacted domain, we collect its geolocation using the GeoIP2 City database.[11] We collect the daily energy mix and resulting carbon intensity from *ElectricityMap* [292] and *OurWorldInData* [293]: for countries and regions covered by *ElectricityMap*, we collect the corresponding carbon intensity hourly. As for other countries, we use the latest known carbon intensity from *OurWorldInData*'s database.

5.4 Empirical Measurement Results

We present the results of our crawling sessions. First, we report on the estimated carbon footprint of advertising observed in *Regular* sessions (section 5.4.1) and then extract the carbon overhead introduced by client side header bidding (section 5.4.2). Then, we show consenting or refusing cookie banners has a significant impact on the carbon footprint of websites (section 5.4.3).

5.4.1 Advertisement on the web

Our results are obtained by crawling $31,394$ pages from $10,562$ domains twice: once with and once without our ad-blocker enabled. For more accuracy, this process is repeated four times, resulting in $230,734$ page visits (when accounting for unresponsive domains or CAPTCHAs). We execute the first three crawls consecutively with the hardware settings described in Section 5.3. To further validate the stability of our approach, we performed a fourth crawl after a period of three months that shows similar results.

In Figure 5.5, we show a log-scale *cumulative distributive function* (CDF) of CO_2eemissions per website for each crawl. It can be observed that our measurements remain stable over time, as the values related to the latest crawl are almost identical to those of the second crawl, albeit some variations can be seen between the first and third crawls.

Table 5.1: Crawled domains per category, including the total number of requests and the ratio of ad-based requests.

Category	**Domains**	**Page visits**	**Requests**	**Ad ratio**
News	1,095	3285	2,970,600	55.18%
Technology	2,268	6804	2,860,591	33.69%
Business	983	2949	1,388,919	33.19%
Entertainment	370	1110	763,048	47.44%
Education	923	2769	1,038,587	31.81%
Shopping	487	1461	910,604	34.41%
Games	272	816	528,996	42.83%
Health	288	864	427,268	42.84%
Finance	369	1107	473,390	34.37%
Others	3,310	9930	4,030,588	33.83%

[11] `https://www.maxmind.com/en/geoip2-city`

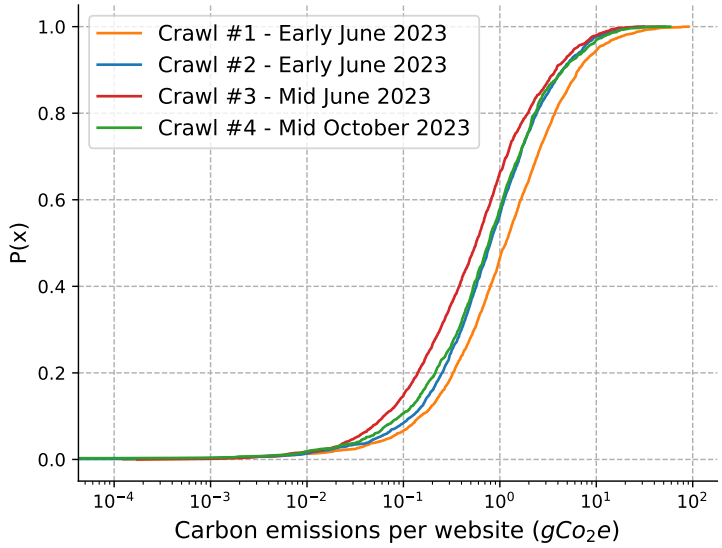


Figure 5.5: CDF of CO_2e emissions per website for each crawl.

On average, we find that 29.94% of requests per website are due to advertising. This number includes both ads and trackers. Figure 5.6 compares the average CO_2eemissions per website when blocking or allowing ads. We observe that 86% of websites emit more when accepting ads, the remaining 14% generated insignificant or undetected advertisement-related network traffic. Additionally, ad-based traffic from 38.15% of websites doubled their CO_2eemissions. A share of websites emitted over $1,000\%$ more when accounting for their ads. We discuss the implications of these results in Section 5.5.

We collected each domain's category through *Symantec's WebPulse Site Review* [294] database. Table 5.1 shows that a significant share of requests belong to advertising. The *News* group has the highest ratio of ad-based requests (over 55%), which is likely due to many news websites relying extensively on ad revenue [295]. Figure 5.7 presents the CO_2eemissions per category. Ads from the *News* group represent the biggest share: over 7.91 $kgCO_2e$ for ad-related content. Categories, such as *Games*, emitted among the lowest ratio of ad-based $CO_2e$with 0.86 $kgCO_2e$, representing 34.76% of the total emissions of the crawls for this group. One explanation is that the gaming industry generates most of its revenue through game sales, microtransactions, or in-game ads [296], which we did not measure since they are not found on their websites.

Cost of advertisement per country We looked at the geographical distribution of the advertising domains. For consistency, we chose domains that perform more than 200 requests (more than 91% of the domains in our dataset meet this criterion). We find that, on average, 15.35% of total ad domains were located in the US, closely followed by Russia (13.48%) and France (12.56%). This ranking correlates with the popularity of domains on TRANCO. Figure 5.8 shows the results for the Top 10 countries with the most requests and their respective CO_2eemissions. The carbon footprint for US-based servers represents 11.17% of the total carbon

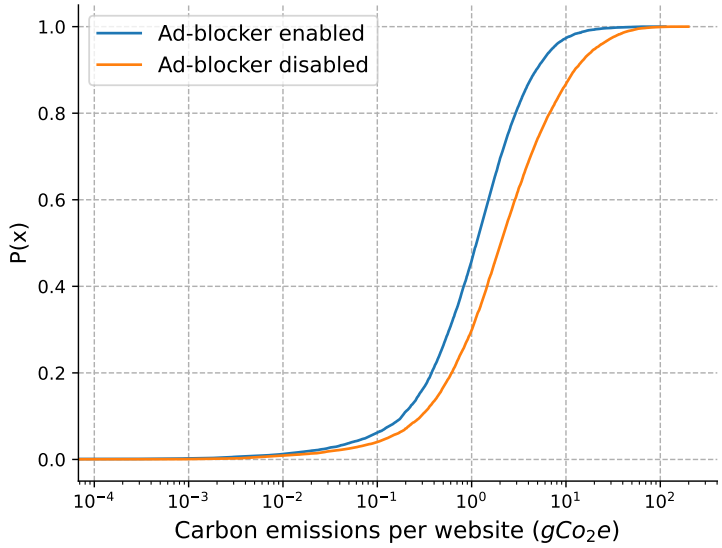


Figure 5.6: CDF of CO_2e emissions (log-scale X-axis)

emissions of our crawls, putting the US in 5th position for highest CO_2eemitters. However, many of the data centers present on US soil are affiliated with major technology firms [297] that claim to be moving their data centers towards greener sources of energy that emit less CO_2e. Since we were unable to verify such claims or obtain specific energy mix values, we did not adjust the data from *ElectricityMap*[12], potentially overestimating the impact for some data centers. France and Russia are second and third in emissions, with 13.03 % and 12.03%, respectively. And Hungary takes the lead as the highest CO_2eemitter with 15.35% of the CO_2eshare, despite handling 11.47% of all advertisement requests.

Carbon emissions per resource type Figure 5.9 presents the total carbon footprint per resource type. As expected, scripts make up the majority of the CO_2eemissions of our crawls. With $4,614,562$ requested scripts in total ($1,862,395$ flagged as an advertisement or tracking-related), they emitted a total of 41.36 $kgCO_2e$ during our crawls (18.71 $kgCO_2e$ for advertisement or tracking-related scripts). Images come second in terms of advertising-related CO_2eemissions, with a total of 1.26 $kgCO_2e$ emitted for advertising purposes. This can be expected as images are the most common form of visual advertisement detected in our crawls with over $1,989,028$ requests, representing 34.57% of the total displayed images. We estimated the footprint of a single image to be $6.35 \cdot 10^{-4} gCO_2e$, while a single script emits an average of $1.00 \cdot 10^{-2} gCO_2eq$. This difference is explained by the fact that most scripts cause more activity on a web page (by initiating new requests or executing resource-consuming instructions), while images require less CPU time to be decoded and rasterized, most of the time. Furthermore, for each request, we collected whether it was, or wasn't, initiated by an advertisement-related script. In total, our crawls emitted 68.34 $kgCO_2e$ when the ad-blocker was disabled. However, 11.74 $kgCO_2e$ could have been saved if advertisement-related scripts did not

[12] `https://electricymaps.net`

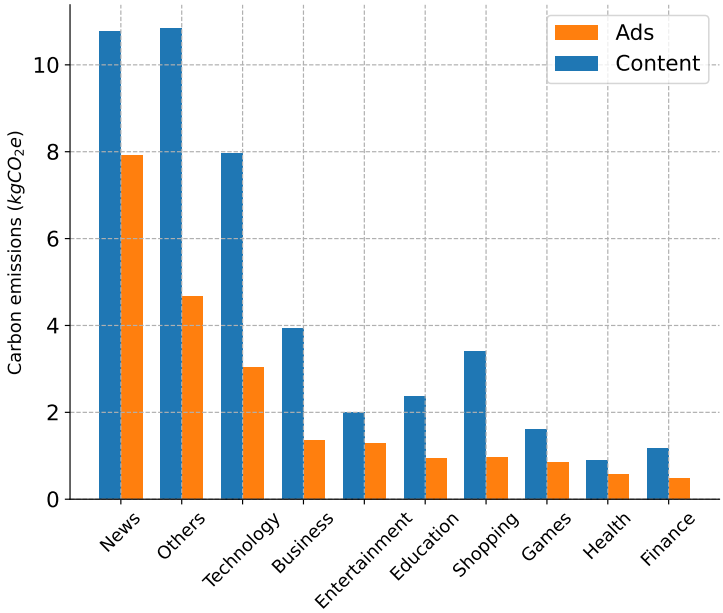


Figure 5.7: CO_2e emissions per category

request multiple additional resources. Finally, out of $47,427$ video media requests, only 13.75% were related to advertisements with the emission of 2.89 gCO_2e for advertisement purposes during our crawls. This low number compared to the other types can be explained by the fact that our crawls did not trigger many video advertisements, as they potentially require direct interaction.

5.4.2 Overhead of header bidding

We have crawled $10,535$ domains and found that $1,098$ of them included *prebid.js*. After filtering every *auctionInit* event, we identified auctions on 845 domains and $1,863$ pages from over 463 bidders. We then identified the requests associated with the bids by looking for the *auction* endpoint or by matching the URLs with the bidders' known domains and auction-related keywords. For all domains (regardless of whether they include *prebid.js*), we further analyzed the content of each network request for occurrences of auction-related keywords and found an additional 309 domains that initiated auctions without including *prebid.js*. We compute the carbon emissions of each bid using the formula from Section 5.3.3.

Through the described methodology, we identified over $125,215$ requests associated with the bidding process. We find that, on average, the bidding process represents an increase of 3.5% of carbon emissions, with an additional 0.16 gCO_2e emitted per domain. We note that only 7 domains saw an increase of over 40% of their gCO_2e emissions due to header bidding. In total, we estimate that the bidding process has emitted over 190 gCO_2e during our crawls. While the carbon footprint's increase due to the bidding process is not significant, we assume that these numbers propose a lower bound on the real emissions of the bidding process, as some auction-related events are harder to identify and might have been missed by our script. In particular,

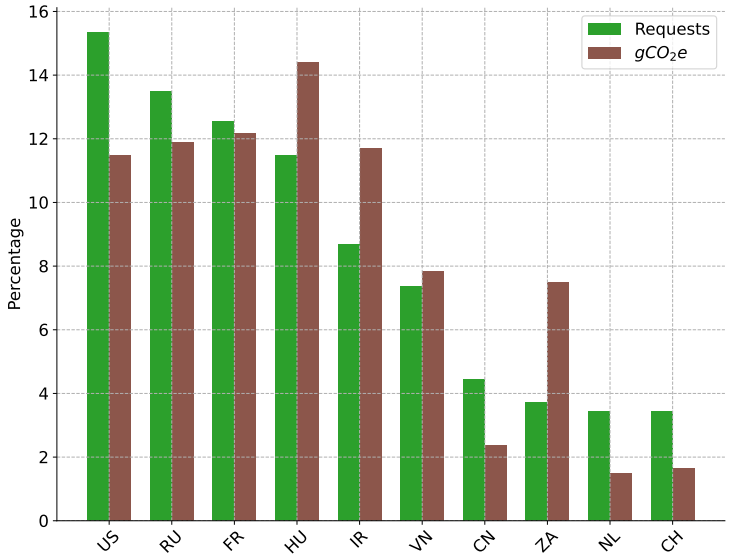


Figure 5.8: Requests & CO_2e emissions per country

our carbon model optimistically assumes that only one server is involved in the bidding process, which might not be the case in practice.

5.4.3 Impact of cookie banners

As stated in Section 5.3, during the *Cookies* crawling session, we visited domains on which we detected a cookie banner in the *Regular* crawling session. We detected $4,052$ such domains, of which $3,608$ were responsive at the time of our *Cookies* session. We crawled each domain with a clean profile, twice: once accepting the cookie banner and once without interacting with it. Figure 5.10 shows the CDF of CO_2eemissions of individual domains when accepting, or ignoring the cookie banner. On average, each website emits 0.37 more gCO_2e when cookies are accepted, with multiple domains emitting over 10 more gCO_2e. In addition, we observed that accepting the cookie banner resulted in an average of 128 additional requests, a 38% increase. More precisely, 104 of those requests are related to advertisements. Additionally, of the $3,608$ responsive websites, 38.8% saw an increase of over 50% of their network requests once we allowed all cookies. In total, our crawls emitted 1.50 $kgCO_2e$ when ignoring cookie banners in contrast to 2.85 $kgCO_2e$ when consenting. Advertisement accounted for 0.39 $kgCO_2e$ and 1.02 $kgCO_2e$, respectively. As the difference is significant, it outlines that websites respecting user consent due to the GDPR or GDPR like legislation generate less network activity, resulting in lower use of the client's resources and a lower toll on the data centers.

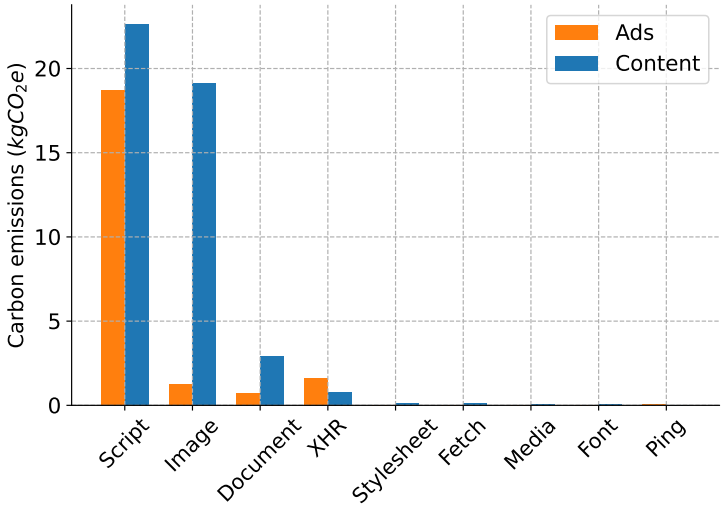


Figure 5.9: CO_2e emissions per resource type

5.5 Discussion

5.5.1 Implication of measurements

Our study shows that ads significantly impact the carbon footprint of a Web page, with, on average, 29.94% of requests per website related to advertising, and as high as 90% for some domains in our dataset. At a time when energy savings are critical, it is important to understand the hidden costs of ads on the Web. Since ads are essential to the Web's economy, changes have important implications, but progress should be made to limit their carbon footprint and understand that the costs are shared, end-to-end. One optimization might be to adjust ads around peak times, when the electric grid is reaching peak energy consumption, ad usage could be throttled to limit its impact: the resolution of video ads could be lowered or replaced by static images, bidding prices on ads could increase so websites display fewer ads without sacrificing revenue, tracking scripts could be simplified or removed to be less resource intensive, cookie syncing could be paused, etc. Interestingly, some sources [298] argue that targeted ads reduce emissions by reducing pointless impressions. Considering the cost of tracking and profile users, running all of the scripts, as well as the fact that websites fill in the ad space with cheaper ads, we feel it's highly unlikely that this is true. They might reduce the cost of an ad campaign, but they're unlikely to reduce the cost of the online advertising ecosystem. Finally, our measurements should not be taken out of context. They are averaged over our crawls, run in controlled environments, using models to estimate consumption, and are better interpreted through ranges and proportions than exact values of CO_2e.

5.5.2 Evolution of the Internet

We see that, thanks to the General Data Protection Reglement (GDPR), not consenting to cookie banners can significantly reduce our carbon footprint. GDPR-like legislation in other countries would help to avoid some of the environmental impact

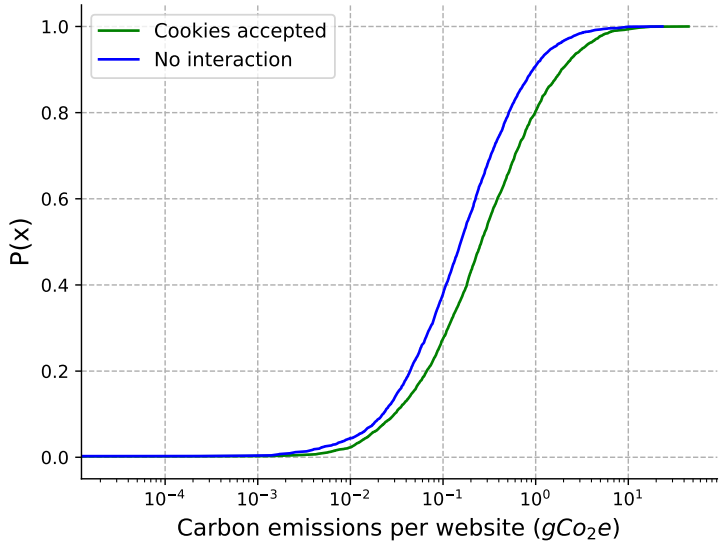


Figure 5.10: CO_2e additional emissions from accepting cookie banners (log-scale X-axis)

of advertising, profiling and ad-based tracking. The use of dark patterns [299] to obtain consent unwillingly, as well as the fatigue that leads to consenting for quick access to the page, should be more closely addressed by researchers and legislators. It's worth mentioning that Google has announced that third-party cookies will be retired in 2024 [300].Client-side header bidding will be directly affected as cookies are sent directly to ad exchanges. Moreover, such a major change will impact studies like this one and others focused measuring ads, tracking or energy consumption on the Web. Nevertheless, at the time of writing, not a single alternative currently being discussed by Google, Mozilla, Apple or Meta has been designed with reduced energy consumption in mind. Current proposals focus on privacy and how advertisers can learn enough about users without being "too privacy invasive", but none seem to be considering the carbon footprint as a primary challenge to be solved. It takes a decade for a change of this magnitude to move from design to ubiquity, we hope that the opportunity to design solutions that reduce the carbon footprint of online advertising is not missed.

5.5.3 Limitations

Although we performed our measurements as realistically as possible, the numbers we present in this study are the result of measurements and estimates based on our model. For data centers, we estimated the cost of processing data but it is known that optimized data center architectures can lead to improvements in energy consumption [301]. As established in the literature, we considered the power consumed by a single router to be a function of the network traffic but various other factors could impact the power consumption. Finally, our study only initiated crawls from one location. We acknowledge that the carbon emissions of a network may vary depending on the source location, the destination, and the route taken by the request, as does the energy mix. Therefore, we suggest that a multi-location

study should be conducted to obtain a more accurate understanding of the carbon footprint of the advertising industry.

Chapter 6

Conclusions

Since its creation by Tim Berners-Lee, the Web has evolved and grown to become ingrained in our everyday lives. Today's Web is a platform for everyone: children get access and are accustomed to the Web at a young age, many aspects of their education is also moving towards the Web, adults use the Web for leisure, work, and hobbies, and the older generation, for whom the Web arrived later in life, benefit from the many aspects it has to offer. One of the reasons why the Web is getting more popular every year can be attributed to the democratization of devices that provide access to the Internet: smartphones are increasingly more accessible and more powerful. Smartwears, such as smartwatches, are also capable of bringing the Internet to our wrists. Smart appliances, such as smart fridges or smart TVs connect our homes to the Internet and provide a constant interface with the Web.

This constantly growing pool of users, as well as the increasing time we spend on the Web, provides the online advertising industry with a constant source of revenue: advertisers distribute their ads everywhere, at any time, all the time, and tailor their ads to individual users through programmatic advertising. This also translates to a quick and steady growth in revenue, which further entices advertising companies to focus their efforts on improved and novel tracking techniques to more extensively profile users and tailor ads, despite increased scrutiny on the associated privacy risks. At a time when user awareness of privacy is rising and regulations are introduced to ensure that Web tracking remains limited, governments are also growing increasingly suspicious of total anonymity on the Web. One such example can be found in the *United-Kingdom*, where the government has attempted to pass a regulation destined to prevent end-to-end encryption under the UL's Online Saftey Bill.[1] This is a rehashing of many such arguments and discussions against anonymity from the 90s,[2] but it is interesting that some of these arguments are regaining traction in public discourse.

Through Web browsers, users have the opportunity to access all types of content on the Internet. In 25 years, browsers evolved from being limited to displaying text to providing features that perform a wide range of operations, such as playing 3D games, watching *virtual-reality* videos, or performing 3D-modelling. All this is made possible by the constant availability of new and improved APIs, such as the WebGL API, the WebVR API, or the Canvas API. However, the demand for new features also provides a growing attack surface for advertisers, with various APIs that can be exploited to track users. Browser fingerprinting is an example of misuse of the diversity of existing browser APIs. As third-party cookies, on which advertisers currently rely for cross-site tracking, are expected to be deprecated in Chrome

[1] https://www.privateinternetaccess.com/blog/uk-leads-the-charge-against-end-to-end-encryption-calls-on-tech-companies-to-nerd-harder/

[2] https://www.nytimes.com/1996/12/19/business/judge-rules-against-us-in-encryption-case.html

this year (2024)[3], the use of stateless tracking techniques and forms of browser fingerprinting are, arguably, set to grow. Along those lines, users expect faster, more powerful, more innovative experiences on the Web, which pushes browser developers to maintain a regular flow of new APIs, which can further enrich browser fingerprints. Despite the transient nature of a browser fingerprint, APIs that provide access to hardware features, such as WebGL, can be exploited to make browser fingerprints contenders to replace third-party cookies for Web tracking.

The advent of smartphones and smart wear prompts children to be in close contact with the Web from a young age [302]. Attracted by a world of easily accessible video-games, and child-oriented content, combined with the ease of use of today's devices, children and young Internet users are spending more and more time accessing the Web. This highly vulnerable pool of population represents significant revenue sources for advertisers: most apps on the App Store for the Apple ecosystem and the Play Store for the Android ecosystem are available for free, driven by an ad-oriented business model. Games are very representative of this model and are the main target for younger users. However, due to their business model, and particularly when offered for free, these apps present serious privacy risks, with various studies, including our work, assessing the presence of a high number of trackers and advertising when compared to their paid counterparts. This is especially worrying knowing their target population. Despite the existence of regulations and policies to protect children, unscrupulous Web developers are taking advantage of the lack of strict enforcement to continue their practices.

Thankfully, both the Web community and ruling bodies have introduced regulations and rules to limit the privacy impact of the online advertising community. Adblockers, such as uBlock Origin, are readily available on browsers and provide serious protection against tracking resources. Browsers that are entirely oriented toward privacy and anonymity, such as the Tor Browser, remain a very good alternative to limit the privacy intrusions on the Web. Upcoming regulations, in the form of the ePrivacy Regulation, are expected to introduce stricter rules toward non-consented data collection. Finally, due to the outcry of the community and raising awareness, even advertisers are seeking potentially more privacy-respecting measures. This last point is important, as advertising is deeply ingrained in today's Web: it is here to stay, and seeking privacy-respecting alternatives should be the priority of the various concerned entities.

At a time when the effects of climate change are more visible, Internet usage is peaking each year, leading to more data being created and distributed, prompting the need for upgrading and building more infrastructure. The online advertising industry is a key contributor to global infrastructure use: through programmatic advertising, algorithms are constantly solicited to identify the most tailored ad for the requesting user, as fast as possible. Environmental issues are currently being scrutinized, with data centers adopting more environmentally friendly cooling systems and optimized devices are being developed with the objective of reducing their *Power Usage Efficiency (PUE)*.

In conclusion, from the creation of the web by Tim Berners-Lee to its current constant presence in our lives, its evolution has been characterized by the democratization of access through a wide range of smart devices. This accessibility has

[3] `https://developers.google.com/privacy-sandbox/blog/cookie-countdown-2023oct`

fuelled the growth of the online advertising industry, which benefits from the constant stream of users and their increasing time spent online. However, privacy concerns and regulatory scrutiny have prompted discussion and action to better protect user data. Despite these challenges, the web continues to evolve, offering innovative experiences through advances in browser technologies and APIs. As we examine the complexity of the tracking and online advertising ecosystem and its environmental impact, it becomes increasingly important for major companies like GAFAM to prioritize privacy policies and sustainable practices to ensure a safer web for all users. In this thesis, I provide an overview of the tracking impact of the online advertising industry by revealing a stable and novel hardware-based attribute, investigating the embedded tracking in the Android gaming ecosystem, and providing a global overview and the first end-to-end measurement of the carbon impact of the advertising ecosystem.

6.1 Contributions

6.1.1 Drawnapart

In Chapter 3, I introduced an effective technique to create a browser fingerprint that relies on minor manufacturing variations in GPUs. To date, this is the first work that exploits hardware accelerated APIs to challenge privacy in this context by successfully distinguishing between identical devices. Our fingerprinting technique can tell apart devices that are otherwise completely indistinguishable by current state-of-the-art methods, while remaining robust to changing environmental conditions. Our technique works well on both desktops and mobile devices, has a practical offline and online runtime, and does not require access to any extra permissions or sensors, such as the microphone, camera, or gyroscope.

Since its introduction in 2011, the WebGL API has been shown to be vulnerable to multiple fingerprinting techniques, such as the Canvas Fingerprinting or through the constants provided by the API. Through DrawnApart, this fingerprinting potential is pushed even farther, showcasing the fingerprinting potential of APIs that empower the browsers' capabilities.

Due to the physical constraints on power consumption and speed of existing processors and materials, modern designs are increasingly relying on massively parallel architectures to improve performance. At the same time, as the capabilities of GPU hardware become increasingly exposed to untrusted Web applications through APIs, such as WebGPU, hardware and software designers must be aware of the risks to privacy raised by hardware fingerprinting, and take care to design software, drivers and hardware stacks in ways that protect user privacy.

6.1.2 A Privacy Analysis of Games in Android

Games on mobile devices generate revenue in different ways: by showing in-game ads, by offering in-app purchases, or by having an up-front cost on the store. We investigated how these economic models can impact user tracking by analyzing the trackers present in 6,355 free and 396 paid mobile games. Overall, we found that free games have on average 3.4 times more trackers than the studied paid games, and they request twice as many dangerous permissions. While the main trackers

in free games are for advertisement purposes, analytics are the most prominent trackers in paid games. We also look at games aimed at a younger audience with the "Educational" game category and the presence of a "Teacher Approved" badge. We find that the stricter policies imposed by Google have had a positive effect on tracking as there are fewer trackers in these games than in the other studied categories.

6.1.3 Environmental Impact of Online Advertising

The advertising industry makes up a significant share of Internet traffic due to its prevalence on the Web. In this chapter, we analyzed the carbon footprint of advertisements by crawling over $31,394$ pages and $10,562$ domains. Overall, we found that advertisements were responsible for an increase of over 144% of the carbon footprints of our crawls, with a total of $68.34\ kgCO_2e$ emitted when allowing advertisements. We also looked at the increased carbon footprint generated by cookie banners and found that accepting all cookies emitted 90% more gCO_2e and 38.8% more advertisement requests on average. Finally, through an analysis of header-bidding processes, we discovered that advertising auctions emit a negligible amount of additional gCO_2e (on the client's device), leading us to believe that the associated carbon cost of header bidding is mostly situated on the servers.

6.2 Future work

6.2.1 Automating the Investigation of Cookie Banners

Google's announcement of the deprecation of third-party cookies in Chromium-based browsers triggered various reactions in the tracking and advertising ecosystem, leading the different actors to look for different tracking techniques that provide a similar tracking efficiency. While third-party cookies reigned in the online tracking industry, cookies banners became widespread, due to the RGPD and other privacy regulations' impulse. The deprecation of third-party cookies might seem like cookie banners will not be as required as previously, however, I believe that their usage will remain widespread, as the industry is strongly incentivized to move to server-side tracking, which leverages first-party cookies. One example of such tracking technique is found in Meta's Pixel API, as explained by Bekos *et al.* [39]. In their study, Bekos *et al.* find that Meta is capable of overcharging its Pixel's request with an identifier that is set through first-party cookies, limiting Meta's reliance on third-party cookies.

Therefore, with updated regulations that encompass this new usage of first-party cookies, cookie banners will remain a useful tool to limit the impacts of online tracking. As such, it is important for the research community to have a varied toolset to understand and monitor the privacy implications of the different choices available in cookie banners. Previous works have already explored automating the interaction with cookie banners: Khandelwal *et al.* [303] have developed *CookieEnforcer*, an automated framework that leverages a T5 model to disable all non-essential cookies. Gundelach *et al.* [304] proposed *Cookiescanner*, which identifies the color difference in buttons to identify the option for declining cookies. Nouwens *et al.* [305] developed *Consent-O-Matic*, a browser extension to select specific options in cookie

banners. Using their tool, they perform a large-scale study of $10,000$ websites to identify dark patterns and measure CMP's compliance with existing regulations.

However, these tools present one major flaw: due to the dynamic aspect of websites, they fail to scale to new cookie banner's design. *CookieEnforcer*, which is based on a T5, is trained to refuse all non-necessary cookies and does not provide the users with choices on which type of cookies they want to allow. Furthermore, T5 models are limited by their context size, which can be significantly higher than the token capacity of many LLMs that run on consumer devices. *Consent-O-Matic* is capable of navigating cookie banners and selecting specific options, but it relies on community-provided configuration files, which need verification and are not guaranteed to exist for specific websites and CMPs.

In this future work, I suggest the development of an automated tool that is capable of fully interacting with the cookie banner, by identifying and selecting different options, or by selecting individual third-parties, and analyzing whether the choices are respected by the website. However, such a tool showcases multiple challenges: First, the tool should be capable of comprehending the flow of clicks required to select specific options. For example, the tool should be able to remember previous actions, as cookie banners typically require users to go through multiple steps in order to alter the default selection. It should be able to understand the meaning of the used wording in cookie banners. For this challenge, I suggest that a LLM is appropriate. However, LLMs that are capable of running on typical user hardware mostly have a limited context window, limiting the amount of information that it is capable of handling at once. Such a tool should be able to parse the Privacy Policy of each website and identify cases where the offered choices in cookie banners are not in compliance. Finally, it should be able to track third-party requests and link the different network requests and page events to the sequence of actions that the automated tool performed. Using such a tool, a large-scale study would help unveil and prevent violations of non-consented data sharing.

6.2.2 Assessing the fingerprinting ability of Web hardware APIs

The WebGL API is set to be replaced by the WebGPU API, which is expected to provide better compatibility with newer GPUs, along with faster operations and access to more advanced GPU features.[4] The WebGL API has been shown to be used for fingerprinting using multiple techniques: canvas fingerprinting is an example of such usage. In this thesis, I contributed to unveil the tracking capabilities of WebGL through DrawnApart, showcasing that the WebGL API, by itself, can be used to re-identify devices, but in conjunction with classic browser fingerprinting techniques, is quite effective at doing so.

With the upcoming release of the WebGPU API and the advanced capabilities it introduces compared to the current WebGL API, it is essential to investigate whether its usage allows for it to be used for fingerprinting. Thus, I propose to investigate existing WebGL-based tracking techniques on WebGPU, and assess the presence of potentially identifiable features. In particular, in the future, I plan to investigate whether more advanced capabilities and access to modern APIs can make the GPU

[4] `https://developer.mozilla.org/en-US/docs/Web/API/WebGPU_API`

more prone to be used in browser fingerprinting, by deriving stable, robust, and unique attributes.

Finally, this approach can be extended to the relatively recent WebXR[5], as it interfaces with various hardware, such as Virtual Reality and Augmented Reality gears. Due to its relatively recent release (February 2022), the security implications of WebXR are still not investigated enough and could benefit from an increased focus from the research community: to date, the only work that is closely related to the security of the new WebXR API has been performed by Lee *et al.* [306], in which they investigate how potential attackers can infer the users' inputs by analyzing the VR controller's movements obtained through the WebVR API (predecessor of WebXR).

6.2.3 Exploring the Impact of Browser Configuration on Browser Fingerprints

To date, studies on browser fingerprints have focused on attributes that can be obtained on the user's default configuration. However, different use cases might lead users to alter their configuration through one or multiple parameters, for example, by editing command line switches, by accessing the standard settings from the browser, or by editing experimental flags. Changes to default settings can alter the way that the browser parses the DOM, and it can add or remove attributes and functions from the APIs that the browser exposes , or even change the browser's performance. Furthermore, different versions of browsers, such as Google Chrome or any of its many variants, can be compiled with different settings, also introducing fingerprint-altering changes.

We make multiple observations:

- As a countermeasure to tracking using browser fingerprinting, it's possible that users can alter their fingerprint through browser settings that have little impact on usability. Such settings will not impact the overall user experience but, if successful, should sufficiently alter the fingerprint to make re-identification more difficult.
- On the other side, advertisers might take advantage of the fingerprint evolutions due to parameter changes to discover potentially highly unique attributes. Advertisers might also potentially uncover the different settings used by analyzing specific changes to fingerprints, which can be used to make a fingerprint more unique.

I argue that an automated tool that explores the impact of browser configurations should be developed. Using such a tool, additional awareness can be provided to browser developers on the privacy and security implications of the changes they introduce. Such development tools should increase privacy at development time before the novel functionalities get shipped, popularized, and exploited.

[5] `https://developer.mozilla.org/en-US/docs/Games/Techniques/3D_on_the_web/WebXR`

6.2.4 Establishing new standards for environmental impact measurements

Through AdCarbon, I presented the first end-to-end measurement of the carbon cost of online advertisement at the network request granularity. Our study is based on real measurements from the user's device, and a combination of measurements and heuristics for network and server-side measurements. One of the biggest limitations of our study is the lack of access to the backends of Web services and to routers. I believe that this opacity is the main reason why the field is lacking such end-to-end studies, especially given the timeliness of environmental considerations.

As a solution, I propose to investigate the creation of a standard for computing the carbon cost related to each network request. I envision this proposal as an HTTP header entry that would relay the request's associated carbon footprint. The industry standard should also define a set of precise rules for computing the carbon cost, based on a software power meter, such as SmartWatts [14], and not introduce any significant strain on the host's resources. The standard should also take into account the privacy considerations linked to sharing carbon-cost measurements for each request, which can potentially lead malicious users to infer the hardware configuration of the server or its location.

Therefore, I argue that a set of standards related to the sharing of the environmental impacts of both servers and routers should be established, which in turn will open multiple possibilities for both academia and the industry to study, optimize, and reduce the environmental impact of the Web.

6.3 Future of online advertising

Public outcry against tracking on the Web has led advertisers to question their practices and motivated official bodies to introduce new and stricter regulations. The future of online advertising, and tracking on the Web, is naturally expected to be shaped by such measures. Google, through their Privacy Sandbox initiative, has proposed multiple standards for advertising: using the Topics API, advertisers are able to tailor their ads to specific categories that describe the user's interests. The Protected Audience API enables ad auctions to be performed on the device, much like client-side header bidding. The difference is that it sources profile information directly on the device, reducing the reliance on third-party data. While these measures show that advertisers are looking for alternatives to third-party cookies for profiling users, even these less privacy-intrusive alternatives can be abused and have been shown that, despite their initial vision, they can be used to track users further than their announced scope. For example, the FLoC[6] API was a proposal by Google that was aimed at replacing cookies with a behavioral-based tracking method: an identifier was to be attributed to each browser's history, and would be classified into a *cohort*. The Web community was quick to point out the weaknesses and privacy issues related to the proposal, such as the potential usage of the FLoC API to enrich browser fingerprints.

Despite growing scrutiny on the security of new browser APIs, they remain vulnerable to fingerprinting. Current browser fingerprints typically rely on many attributes to

[6] `https://github.com/WICG/floc`

create an accurate identifier of users. However, through DrawnApart, we have shown that this requirement can be overlooked, and identifying single robust, stable and unique attributes is enough to construct a fingerprint . Thus, with the increased focus on browser fingerprinting as a tracking alternative to third-party cookies, I argue that the amount of attributes used in future fingerprints will decrease, and be replaced by a limited number of stable attributes with high entropy. As shown by Botvinnik *et al.* [307], WebAssembly's support on browsers could open the door to the discovery of robust attributes based on hardware characteristics. The usage of WebAssembly to perform hardware fingerprinting is also showcased in [88], in which the authors adopt a series of benchmarks that leverage Javascript and WebAssembly to fingerprint the CPU. Some of the proposed benchmarks are capable of being exploited by attacker due to their speed, with timings ranging from about 2 seconds for the fastest benchmark.

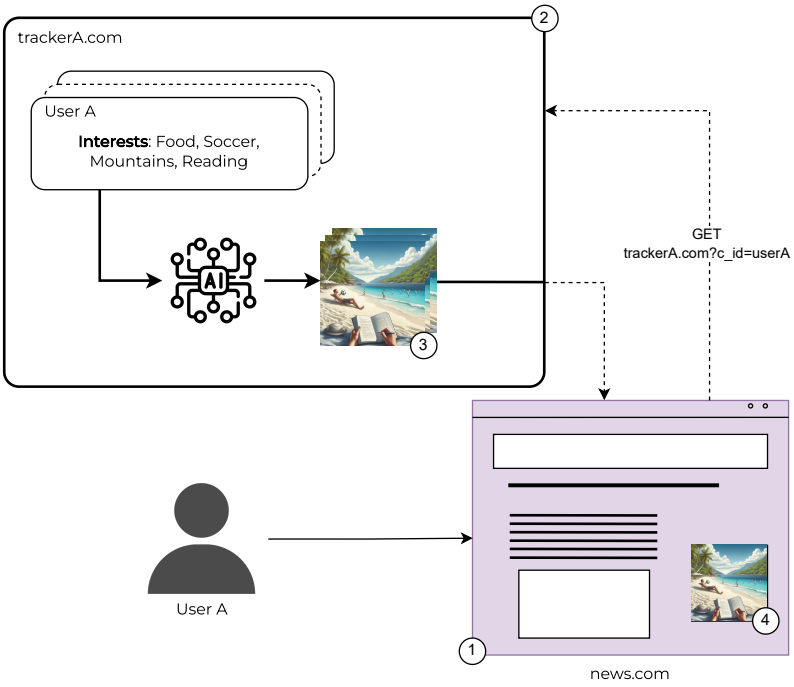


Figure 6.1: An example of a generative AI-based advertising process: (1) The user accesses news.com, which includes scripts belonging to trackerA; (2) The script triggers a request to *trackerA.com* and includes a user-specific identifier; (3) The tracker pulls the user's recorded interests and generates a tailored ad using generative models; (4) The tailored ad is displayed to the user.

Moreover, recent years have seen the explosion of usable *Large Language Models (LLMs)* and more generally, of generative AI, and their capabilities are being used in various fields: arts, code generation or chatbots are examples of such applications. It can be expected that the future of online advertising will be strongly shaped by the evolutions of generative AIs, either during the ad-creation process, with advertisers using the artistic capabilities of such models to lower the costs of ad creation.

Or to shape and generate ads on the fly for the users. For example, while using search engines , existing ads can be manipulated to advertise specific products in their responses based on the user's search queries. One example of such an application has been investigated by Feizi *et al.* [308] in a recent study that shows how LLMs' outputs can be automatically altered to include subtle ads into the generated responses. They propose a framework composed of multiple modules to alter the original responses and initiate an auction process, which allows the advertisers to bid on having their ads included in the altered response. Their framework can be integrated into search engines, such as Bing, which currently leverages LLM-based results The generative AI-supported creative process is already a possibility. Google already includes AI-powered ads processes[7], with the *automatically created assets* for Search Ads, which generates ads headlines based on contextual information. The recently introduced models also offer the possibility for ad creatives to present a tailored version of their ads to individual users: models can expand from a general idea and product to promote and tailor the creative for the likes of the user they are targeting. An example of such a process is depicted in Figure 6.1 However, in order for these models to be efficient, enormous amounts of training data are required, leading to increased data collection. If these types of AI-generated advertisements turn out to be profitable, this might lead advertisers to research and identify more potentially invasive practices to harvest more data, despite ongoing efforts to increase privacy on the Web.

Finally, the explosion of connected devices significantly improves the reach of advertisers. Smart TVs are able to collect information about our entertainment habits, smart fridges are able to keep track of the brands we purchase, and smartwatches are capable of tracking day-to-day activities thanks to the multiple embedded sensors. Advertising on these devices is already present, and advertisers can link and correlate data obtained from multiple devices to provide cross-device advertising. Perhaps one of the changes that will benefit advertisers in the future is their ability to measure the impact of their advertising campaigns through feedback from all the connected devices that are owned by individual users. One of the biggest issues that advertising is currently facing is the reduction of relevant signals regarding their ad campaigns: it is currently difficult to assess whether an ad campaign is the cause of a purchase that occurred days, weeks, or months later. Through the proliferation of smart devices that are able of collecting and analyzing significant amounts of information , advertisers will be able to more thoroughly assess the impact of their ad campaigns, and in consequence, target users more precisely. On the other hand, these benefits to the advertisers come at the expense of privacy. I argue and even hope, that while advertisers will be able to benefit from this pool of data, it is likely to be short-lived before strong regulations and community-based efforts address the privacy risks linked with such practices.

To conclude, my thesis provides a set of contributions related to privacy on the Web, where the underlying theme, often below the surface of my individual contributions, is online advertising. I have studied advanced tracking techniques that may replace third-party cookie tracking, the impact of economic models on tracking in Android apps, as well as the environmental impact of online advertising.

[7] `https://blog.google/products/ads-commerce/ai-powered-ads-google-marketing-live/`

Bibliography

[1] Imane Fouad, Cristiana Santos, Arnaud Legout, and Nataliia Bielova. My cookie is a phoenix: detection, measurement, and lawfulness of cookie respawning with browser fingerprinting. In *PETS 2022-22nd Privacy Enhancing Technologies Symposium*, 2022.

[2] Manifest.permission – Android Developers. `https://developer.android.com/reference/android/Manifest.permission`, 2021.

[3] Somipam R Shimray and Chennupati K Ramaiah. Use of internet through mobile devices: a survey. *SRELS Journal of Information Management*, 56(2):100–105, 2019.

[4] Meg Leta Jones. Cookies: a legacy of controversy. *Internet Histories*, 4(1):87–104, 2020.

[5] Jonathan R Mayer. Any person... a pamphleteer": Internet anonymity in the age of web 2.0. *Undergraduate Senior Thesis, Princeton University*, 85, 2009.

[6] Peter Eckersley. How unique is your web browser? In *Privacy Enhancing Technologies: 10th International Symposium, PETS 2010, Berlin, Germany, July 21-23, 2010. Proceedings 10*, pages 1–18. Springer, 2010.

[7] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 878–894. IEEE, 2016.

[8] Pierre Laperdrix, Nataliia Bielova, Benoit Baudry, and Gildas Avoine. Browser fingerprinting: A survey. *ACM Transactions on the Web (TWEB)*, 14(2):1–33, 2020.

[9] Alejandro Gómez-Boix, Pierre Laperdrix, and Benoit Baudry. Hiding in the crowd: an analysis of the effectiveness of browser fingerprinting at large scale. In *Proceedings of the 2018 world wide web conference*, pages 309–318, 2018.

[10] Brian Dean. Ad blocker usage and demographic statistics in 2023. `https://backlinko.com/ad-blockers-users`, 2021.

[11] Ha Dao and Kensuke Fukuda. Characterizing cname cloaking-based tracking on the web. In *TMA*, 2020.

[12] Antoine Vastel, Pierre Laperdrix, Walter Rudametkin, and Romain Rouvoy. Fp-stalker: Tracking browser fingerprint evolutions. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 728–741. IEEE, 2018.

[13] Cody Taylor and Jonathan Koomey. Estimating energy use and greenhouse gas emissions of internet advertising. *Network*, 2008.

[14] Guillaume Fieni, Romain Rouvoy, and Lionel Seinturier. Smartwatts: Self-calibrating software-defined power meter for containers. In *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pages 479–488. IEEE, 2020.

[15] Tomer Laor, Naif Mehanna, Antonin Durey, Vitaly Dyadyuk, Pierre Laperdrix, Clémentine Maurice, Yossi Oren, Romain Rouvoy, Walter Rudametkin, and Yuval Yarom. Drawnapart: A device identification technique based on remote

gpu fingerprinting. In *Network and Distributed System Security Symposium*, 2022.

[16] Pierre Laperdrix, Naif Mehanna, Antonin Durey, and Walter Rudametkin. The price to play: a privacy analysis of free and paid games in the android ecosystem. In *Proceedings of the ACM Web Conference 2022*, pages 3440–3449, 2022.

[17] Naif Mehanna and Walter Rudametkin. Caught in the game: On the history and evolution of web browser gaming. In *Companion Proceedings of the ACM Web Conference 2023*, WWW '23 Companion, page 601–609, New York, NY, USA, 2023. Association for Computing Machinery.

[18] Naif Mehanna, Walter Rudametkin, Pierre Laperdrix, and Antoine Vastel. Free Proxies Unmasked: A Vulnerability and Longitudinal Analysis of Free Proxy Services. In *Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb'24)*, San Diego (CA), United States, February 2024.

[19] Rémi Després. X. 25 virtual circuits–transpac in france–pre-internet data networking. *IEEE Communications magazine*, 48(11):40, 2010.

[20] William L Cats-Baril and Tawfik Jelassi. The french videotex system minitel: A successful implementation of a national information technology infrastructure. *Mis Quarterly*, pages 1–20, 1994.

[21] Drew Dean, Edward W Felten, Dan S Wallach, Dirk Balfanz, and PJ Denning. Java security: Web browsers and beyond. *Internet besieged: countering cyberspace scofflaws*, pages 241–269, 1997.

[22] Flavio De Paoli, Andre L Dos Santos, and Richard A Kemmerer. Web browsers and security. *Mobile Agents and Security*, pages 235–256, 1998.

[23] Edward W Felten and Michael A Schneider. Timing attacks on web privacy. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*, pages 25–32, 2000.

[24] Jonathan Tamary and Dror G Feitelson. The rise of chrome. *PeerJ Computer Science*, 1:e28, 2015.

[25] Nicholas Carlini, Adrienne Porter Felt, and David Wagner. An evaluation of the google chrome extension security architecture. In *21st USENIX Security Symposium (USENIX Security 12)*, pages 97–111, 2012.

[26] Alan J Reid and Alan J Reid. A brief history of the smartphone. *The Smartphone Paradox: Our Ruinous Dependency in the Device Age*, pages 35–66, 2018.

[27] Ernie Dainow. *A concise history of computers, smartphones and the internet*. Ernie Dainow, 2017.

[28] Apple. App store review guidelines. `https://developer.apple.com/app-store/review/guidelines/`, 2023.

[29] Apple. Apple developer program. `https://developer.apple.com/programs/`, 2023.

[30] Google. How to use play console. `https://support.google.com/googleplay/android-developer/answer/6112435`, 2023.

[31] Statista. Global market share held by mobile operating systems from 2009 to 2023. `https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/`, 2023.

[32] MICHELLE FAVERIO and MONICA ANDERSON. For shopping, phones are common and influencers have become a factor–especially for young adults. 2022.

[33] Laura Silver, Aaron Smith, Courtney Johnson, Jingjing Jiang, Monica Anderson, and Lee Rainie. Use of smartphones and social media is common across most emerging economies. *Pew Research Center*, 7, 2019.

[34] Dean Donaldson. Online advertising history. *London: Bournemouth Media School*, 2008.

[35] Daniel Belanche. Ethical limits to the intrusiveness of online advertising formats: A critical review of better ads standards. *Journal of marketing communications*, 25(7):685–701, 2019.

[36] Hairong Li, Steven M Edwards, and Joo-Hyun Lee. Measuring the intrusiveness of advertisements: Scale development and validation. *Journal of advertising*, 31(2):37–47, 2002.

[37] Tim Jackson. This bug in your pc is a smart cookie. `https://archive.org/details/FinancialTimes1996UKEnglish/Feb%2012%201996%2C%20Financial%20Times%2C%20%2312%2C%20UK%20%28en%29/page/n29/mode/2up`, 1996.

[38] Tereza Semerádová and Petr Weinlich. Computer estimation of customer similarity with facebook lookalikes: Advantages and disadvantages of hypertargeting. *IEEE Access*, 7:153365–153377, 2019.

[39] Paschalis Bekos, Panagiotis Papadopoulos, Evangelos P Markatos, and Nicolas Kourtellis. The hitchhiker's guide to facebook web tracking with invisible pixels and click ids. In *Proceedings of the ACM Web Conference 2023*, pages 2132–2143, 2023.

[40] Shunyao Yan, Klaus M Miller, and Bernd Skiera. How does the adoption of ad blockers affect news consumption? *Journal of Marketing Research*, 59(5):1002–1018, 2022.

[41] Garrett A Johnson, Scott K Shriver, and Shaoyin Du. Consumer privacy choice in online advertising: Who opts out and at what cost to industry? *Marketing Science*, 39(1):33–51, 2020.

[42] Statista. Mobile advertising spending worldwide 20072024. `https://www.statista.com/statistics/303817/mobile-internet-advertising-revenue-worldwide/`, 2022.

[43] Statista. Digital advertising worldwide. `https://www.statista.com/outlook/dmo/digital-advertising/worldwide#ad-spending`, 2023.

[44] Intergovernmental Panel On Climate Change. Climate change 2007: The physical science basis. *Agenda*, 6(07):333, 2007.

[45] Susan Solomon. The mystery of the antarctic ozone "hole". *Reviews of Geophysics*, 26(1):131–148, 1988.

[46] Flavio Scrucca, Grazia Barberio, Valentina Fantin, Pier Luigi Porta, and Marco Barbanera. Carbon footprint: Concept, methodology and calculation. *Carbon Footprint Case Studies: Municipal Solid Waste Management, Sustainable Road Transport and Carbon Sequestration*, pages 1–31, 2021.

[47] Publicly Available Specification et al. Specification for the assessment of the life cycle greenhouse gas emissions of goods and services. *Bsi Br. Stand. Isbn*, 978:580, 2008.

[48] Divya Pandey, Madhoolika Agrawal, and Jai Shanker Pandey. Carbon footprint: current methods of estimation. *Environmental monitoring and assessment*, 178:135–160, 2011.

[49] Thomas Wiedmann and Jan Minx. A definition of 'carbon footprint'. *Ecological economics research trends*, 1(2008):1–11, 2008.

[50] Bruno Notarnicola, Giuseppe Tassielli, Pietro A Renzulli, and Agata Lo Giudice. Life cycle assessment in the agri-food sector: An overview of its key aspects, international initiatives, certification, labelling schemesand methodological issues. *Life Cycle Assessment in the Agri-food Sector: Case Studies, Methodological Issues and Best Practices*, pages 1–56, 2015.

[51] David Andersson. A novel approach to calculate individuals' carbon footprints using financial transaction data–app development and design. *Journal of Cleaner Production*, 256:120396, 2020.

[52] Matthias Finkbeiner. Carbon footprinting—opportunities and threats, 2009.

[53] J Paul Padgett, Anne C Steinemann, James H Clarke, and Michael P Vandenbergh. A comparison of carbon calculators. *Environmental impact assessment review*, 28(2-3):106–115, 2008.

[54] John Mulrow, Katherine Machaj, Joshua Deanes, and Sybil Derrible. The state of carbon footprint calculators: An evaluation of calculator design and user interaction features. *Sustainable Production and Consumption*, 18:33–40, 2019.

[55] Bertil Vilhelmson, Erik Elldér, and Eva Thulin. What did we do when the internet wasn't around? variation in free-time activities among three young-adult cohorts from 1990/1991, 2000/2001, and 2010/2011. *New Media & Society*, 20(8):2898–2916, 2018.

[56] Jeffrey A Hall and Dong Liu. Social media use, social displacement, and well-being. *Current Opinion in Psychology*, 46:101339, 2022.

[57] Jean M Twenge, Gabrielle N Martin, and Brian H Spitzberg. Trends in us adolescents' media use, 1976–2016: The rise of digital media, the decline of tv, and the (near) demise of print. *Psychology of Popular Media Culture*, 8(4):329, 2019.

[58] Kieren Mayers, Jonathan Koomey, Rebecca Hall, Maria Bauer, Chris France, and Amanda Webb. The carbon footprint of games distribution, 2015.

[59] Altanshagai Batmunkh. Carbon footprint of the most popular social media platforms. *Sustainability*, 14(4):2195, 2022.

[60] Elisabeth Hochschorner, György Dán, and Åsa Moberg. Carbon footprint of movie distribution via the internet: a swedish case study. *Journal of Cleaner Production*, 87:197–207, 2015.

[61] Michael Z Hauschild, Ralph K Rosenbaum, and Stig Irving Olsen. *Life cycle assessment*, volume 2018. Springer, 2018.

[62] Lotfi Belkhir and Ahmed Elmeligi. Assessing ict global emissions footprint: Trends to 2040 & recommendations. *Journal of cleaner production*, 177:448–463, 2018.

[63] Gergely Biczók, Jens Malmodin, and Albrecht Fehske. Economic and ecological impact of ict. *EARTH Project*, 2011.

[64] U Cisco. Cisco annual internet report (2018–2023) white paper. *Cisco: San Jose, CA, USA*, 10(1):1–35, 2020.

[65] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 1388–1401, New York, NY, USA, 2016. Association for Computing Machinery.

[66] Narseo Vallina-Rodriguez, Jay Shah, Alessandro Finamore, Yan Grunenberger, Konstantina Papagiannaki, Hamed Haddadi, and Jon Crowcroft. Breaking for commercials: characterizing mobile advertising. In *Proceedings of the 2012 Internet measurement conference*, pages 343–356, 2012.

[67] 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*, pages 289–299, 2015.

[68] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. Detecting and defending against {Third-Party} tracking on the web. In *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 155–168, 2012.

[69] 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*, pages 674–689, 2014.

[70] Jonathan R Mayer and John C Mitchell. Third-party web tracking: Policy and technology. In *2012 IEEE symposium on security and privacy*, pages 413–427. IEEE, 2012.

[71] World Wide Web Consortium. Same origin policy. `https://www.w3.org/Security/wiki/Same_Origin_Policy`, 2010.

[72] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos Markatos. Cookie synchronization: Everything you always wanted to know but were afraid to ask. In *The World Wide Web Conference*, pages 1432–1442, 2019.

[73] Ashkan Soltani, Shannon Canty, Quentin Mayo, Lauren Thomas, and Chris Jay Hoofnagle. Flash cookies and privacy. In *2010 AAAI Spring Symposium Series*, 2010.

[74] Mika D Ayenson, Dietrich James Wambach, Ashkan Soltani, Nathan Good, and Chris Jay Hoofnagle. Flash cookies and privacy ii: Now with html5 and etag respawning. *Available at SSRN 1898390*, 2011.

[75] Google. Preparing for the end of third-party cookies. `https://developers.google.com/privacy-sandbox/blog/cookie-countdown-2023oct`, 2023.

[76] Nurullah Demir, Daniel Theis, Tobias Urban, and Norbert Pohlmann. Towards understanding first-party cookie tracking in the field. *arXiv preprint arXiv:2202.01498*, 2022.

[77] Quan Chen, Panagiotis Ilia, Michalis Polychronakis, and Alexandros Kapravelos. Cookie swap party: Abusing first-party cookies for web tracking. In *Proceedings of the Web Conference 2021*, pages 2117–2129, 2021.

[78] An updated timeline for Privacy Sandbox milestones – Google. `https://blog.google/products/chrome/updated-timeline-privacy-sandbox-milestones/`, 2021.

[79] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *2013 IEEE Symposium on Security and Privacy*, pages 541–555. IEEE, 2013.

[80] Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel. Fpdetective: dusting the web for fingerprinters. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1129–1140, 2013.

[81] Nasser Mohammed Al-Fannah, Wanpeng Li, and Chris J Mitchell. Beyond cookie monster amnesia: Real world persistent online tracking. In *Information Security: 21st International Conference, ISC 2018, Guildford, UK, September 9–12, 2018, Proceedings 21*, pages 481–501. Springer, 2018.

[82] Umar Iqbal, Steven Englehardt, and Zubair Shafiq. Fingerprinting the fingerprinters: Learning to detect browser fingerprinting behaviors. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1143–1161. IEEE, 2021.

[83] Julian Fietkau, Kashyap Thimmaraju, Felix Kybranz, Sebastian Neef, and Jean-Pierre Seifert. The elephant in the background: A quantitative approachto empower users against web browser fingerprinting. In *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society*, pages 167–180, 2021.

[84] Alexander Sjösten, Daniel Hedin, and Andrei Sabelfeld. Essentialfp: Exposing the essence of browser fingerprinting. In *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 32–48. IEEE, 2021.

[85] Junhua Su and Alexandros Kapravelos. Automatic discovery of emerging browser fingerprinting techniques. In *Proceedings of the ACM Web Conference 2023*, pages 2178–2188, 2023.

[86] Keaton Mowery and Hovav Shacham. Pixel perfect: Fingerprinting canvas in HTML5. *W2SP*, pages 1–12, 2012.

[87] Jordan S Queiroz and Eduardo L Feitosa. A web browser fingerprinting method based on the web audio api. *The Computer Journal*, 62(8):1106–1120, 2019.

[88] Leon Trampert, Christian Rossow, and Michael Schwarz. Browser-based cpu fingerprinting. In *European Symposium on Research in Computer Security*, pages 87–105. Springer, 2022.

[89] Nampoina Andriamilanto, Tristan Allard, Gaëtan Le Guelvouit, and Alexandre Garel. A large-scale empirical analysis of browser fingerprints properties for web authentication. *ACM Transactions on the Web (TWEB)*, 16(1):1–62, 2021.

[90] Antonin Durey, Pierre Laperdrix, Walter Rudametkin, and Romain Rouvoy. Fp-redemption: Studying browser fingerprinting adoption for the sake of web security. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 18th International Conference, DIMVA 2021, Virtual Event, July 14–16, 2021, Proceedings 18*, pages 237–257. Springer, 2021.

[91] Xu Lin, Panagiotis Ilia, Saumya Solanki, and Jason Polakis. Phish in sheep's clothing: Exploring the authentication pitfalls of browser fingerprinting. In

31st USENIX Security Symposium (USENIX Security 22), pages 1651–1668, 2022.

[92] Pierre Laperdrix, Gildas Avoine, Benoit Baudry, and Nick Nikiforakis. Morellian analysis for browsers: Making web authentication stronger with canvas fingerprinting. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 16th International Conference, DIMVA 2019, Gothenburg, Sweden, June 19–20, 2019, Proceedings 16*, pages 43–66. Springer, 2019.

[93] José Estrada-Jiménez, Javier Parra-Arnau, Ana Rodríguez-Hoyos, and Jordi Forné. Online advertising: Analysis of privacy threats and protection approaches. *Computer Communications*, 100:32–51, 2017.

[94] Arno Schäfer and Oliver Weiss. Understanding demand-side-platforms. *Programmatic Advertising: The Successful Transformation to Automated, Data-Driven Marketing in Real-Time*, pages 75–86, 2016.

[95] Tao Lei, Junpeng Gong, and Yujun Wen. Online advertising demand-side platform business system design exploration. In *2015 IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS)*, pages 535–538. IEEE, 2015.

[96] Sam Macbeth. Tracking the trackers: analysing the global tracking landscape with ghostrank. *White Paper of Ghostery*, pages 1–13, 2017.

[97] Giridhari Venkatadri, Piotr Sapiezynski, Elissa M Redmiles, Alan Mislove, Oana Goga, Michelle Mazurek, and Krishna P Gummadi. Auditing offline data brokers via facebook's advertising platform. In *The World Wide Web Conference*, pages 1920–1930, 2019.

[98] Jamie Pinchot, Adnan A Chawdhry, and Karen Paullet. Data privacy issues in the age of data brokerage: an exploratory literature review. *Issues in Information Systems*, 19(3), 2018.

[99] Chih-Liang Yeh. Pursuing consumer empowerment in the age of big data: A comprehensive regulatory framework for data brokers. *Telecommunications Policy*, 42(4):282–292, 2018.

[100] Donald MacKenzie, Koray Caliskan, and Charlotte Rommerskirchen. The longest second: Header bidding and the material politics of online advertising. *Economy and Society*, 52(3):554–578, 2023.

[101] Oliver Busch. The programmatic advertising principle. In *Programmatic advertising: The successful transformation to automated, data-driven marketing in real-time*, pages 3–15. Springer, 2015.

[102] Shanmugavelayutham Muthukrishnan. Ad exchanges: Research issues. In *International workshop on internet and network economics*, pages 1–12. Springer, 2009.

[103] S Muthukrishnan. Internet ad auctions: Insights and directions. In *International Colloquium on Automata, Languages, and Programming*, pages 14–23. Springer, 2008.

[104] Tanmoy Chakraborty, Eyal Even-Dar, Sudipto Guha, Yishay Mansour, and S Muthukrishnan. Selective call out and real time bidding. In *Internet and Network Economics: 6th International Workshop, WINE 2010, Stanford, CA, USA, December 13-17, 2010. Proceedings 6*, pages 145–157. Springer, 2010.

[105] Shuai Yuan, Jun Wang, and Xiaoxue Zhao. Real-time bidding for online advertising: measurement and analysis. In *Proceedings of the seventh international workshop on data mining for online advertising*, pages 1–8, 2013.

[106] What is Header Bidding? `https://www.lotame.com/back-basics-header-bidding/`, 2017.

[107] Michalis Pachilakis, Panagiotis Papadopoulos, Evangelos P Markatos, and Nicolas Kourtellis. No more chasing waterfalls: a measurement study of the header bidding ad-ecosystem. In *Proceedings of the Internet Measurement Conference*, pages 280–293, 2019.

[108] European Commission. General Data Protection Regulation (GDPR). `https://ec.europa.eu/info/law/law-topic/data-protection/eu-data-protection-rules_en`.

[109] Martin Degeling, Christine Utz, Christopher Lentzsch, Henry Hosseini, Florian Schaub, and Thorsten Holz. We value your privacy... now take some cookies: Measuring the gdpr's impact on web privacy. *arXiv preprint arXiv:1808.05096*, 2018.

[110] Thomas Linden, Rishabh Khandelwal, Hamza Harkous, and Kassem Fawaz. The privacy policy landscape after the gdpr. *arXiv preprint arXiv:1809.08396*, 2018.

[111] Iskander Sanchez-Rola, Matteo Dell'Amico, Platon Kotzias, Davide Balzarotti, Leyla Bilge, Pierre-Antoine Vervier, and Igor Santos. Can i opt out yet? gdpr and the global illusion of cookie control. In *Proceedings of the 2019 ACM Asia conference on computer and communications security*, pages 340–351, 2019.

[112] Célestin Matte, Nataliia Bielova, and Cristiana Santos. Do cookie banners respect my choice?: Measuring legal compliance of banners from iab europe's transparency and consent framework. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 791–809. IEEE, 2020.

[113] Karlo Lukic, Klaus M Miller, and Bernd Skiera. The impact of the general data protection regulation (gdpr) on online tracking. *Available at SSRN*, 2023.

[114] Dino Bollinger. Analyzing cookies compliance with the gdpr. Master's thesis, ETH Zurich, 2021.

[115] Elizabeth Liz Harding, Jarno J Vanto, Reece Clark, L Hannah Ji, and Sara C Ainsworth. Understanding the scope and impact of the california consumer privacy act of 2018. *Journal of Data Protection & Privacy*, 2(3):234–253, 2019.

[116] W Gregory Voss. First the gdpr, now the proposed eprivacy regulation. *Journal of Internet Law*, 21(1):3–11, 2017.

[117] Craig E Wills and Mihajlo Zeljkovic. A personalized approach to web privacy: awareness, attitudes and actions. *Information Management & Computer Security*, 19(1):53–73, 2011.

[118] Gaurav Aggarwal, Elie Bursztein, Collin Jackson, and Dan Boneh. An analysis of private browsing modes in modern browsers. In *19th USENIX Security Symposium (USENIX Security 10)*, 2010.

[119] Artur Janc, Krzysztof Kotowicz, Lukas Weichselbaum, and Roberto Clapis. Information leaks via safari's intelligent tracking prevention. *arXiv preprint arXiv:2001.07421*, 2020.

[120] Ahmed Redha Mahlous and Houssam Mahlous. Private browsing forensic analysis: A case study of privacy preservation in the brave browser. *International Journal of Intelligent Engineering & Systems*, 13(6), 2020.

[121] Douglas J Leith. Web browser privacy: What do browsers say when they phone home? *IEEE Access*, 9:41615–41627, 2021.

[122] David M Goldschlag, Michael G Reed, and Paul F Syverson. Hiding routing information. In *International workshop on information hiding*, pages 137–150. Springer, 1996.

[123] Akshaya Mani, T Wilson-Brown, Rob Jansen, Aaron Johnson, and Micah Sherr. Understanding tor usage with privacy-preserving measurement. In *Proceedings of the Internet Measurement Conference 2018*, pages 175–187, 2018.

[124] Abid Khan Jadoon, Waseem Iqbal, Muhammad Faisal Amjad, Hammad Afzal, and Yawar Abbas Bangash. Forensic analysis of tor browser: a case study for privacy and anonymity on the web. *Forensic science international*, 299:59–73, 2019.

[125] Tor Project. The design and implementation of the tor browser. `https://2019.www.torproject.org/projects/torbrowser/design/#fingerprinting-defenses.`, 2019.

[126] Dolière Francis Somé. Empoweb: empowering web applications with browser extensions. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 227–245. IEEE, 2019.

[127] Zainul Abi Din, Panagiotis Tigas, Samuel T King, and Benjamin Livshits. {PERCIVAL}: Making {In-Browser} perceptual ad blocking practical with deep learning. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pages 387–400, 2020.

[128] Florian Tramèr, Pascal Dupré, Gili Rusak, Giancarlo Pellegrino, and Dan Boneh. Adversarial: Perceptual ad blocking meets adversarial machine learning. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 2005–2021, 2019.

[129] Reethika Ramesh, Leonid Evdokimov, Diwen Xue, and Roya Ensafi. Vpnalyzer: systematic investigation of the vpn ecosystem. In *Network and Distributed System Security*, pages 24–28, 2022.

[130] Yana Dimova, Gunes Acar, Lukasz Olejnik, Wouter Joosen, and Tom Van Goethem. The cname of the game: Large-scale analysis of dns-based tracking evasion. *arXiv preprint arXiv:2102.09301*, 2021.

[131] Quan Chen, Peter Snyder, Ben Livshits, and Alexandros Kapravelos. Detecting filter list evasion with event-loop-turn granularity javascript signatures. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1715–1729. IEEE, 2021.

[132] Ward Van Heddeghem, Sofie Lambert, Bart Lannoo, Didier Colle, Mario Pickavet, and Piet Demeester. Trends in worldwide ict electricity consumption from 2007 to 2012. *Computer Communications*, 50:64–76, 2014.

[133] Anders SG Andrae. New perspectives on internet electricity use in 2030. *Engineering and Applied Science Letters*, 3(2):19–31, 2020.

[134] Jayant Baliga, Kerry Hinton, Robert Ayre, and Rodney S Tucker. Carbon footprint of the internet. 2009.

[135] Jens Malmodin, Dag Lundén, Åsa Moberg, Greger Andersson, and Mikael Nilsson. Life cycle assessment of ict: Carbon footprint and operational electricity use from the operator, national, and subscriber perspective in sweden. *Journal of Industrial Ecology*, 18(6):829–845, 2014.

[136] Peter Corcoran and Anders Andrae. Emerging trends in electricity consumption for consumer ict. *National University of Ireland, Galway, Connacht, Ireland, Tech. Rep*, 2013.

[137] Jens Malmodin and Dag Lundén. The energy and carbon footprint of the global ict and e&m sectors 2010–2015. *Sustainability*, 10(9):3027, 2018.

[138] Matti Pärssinen, M Kotila, R Cuevas, A Phansalkar, and Jukka Manner. Environmental impact assessment of online advertising. *Environmental Impact Assessment Review*, 73:177–200, 2018.

[139] José González-Cabañas, Patricia Callejo, Rubén Cuevas, Steffen Svartberg, Tommy Torjesen, Ángel Cuevas, Antonio Pastor, and Mikko Kotila. Carbontag: A browser-based method for approximating energy consumption of online ads. *IEEE Transactions on Sustainable Computing*, 2023.

[140] Antoine Vastel, Pierre Laperdrix, Walter Rudametkin, and Romain Rouvoy. FP-Stalker: tracking browser fingerprint evolutions. In *IEEE SP*, pages 728–741, 2018.

[141] Charles Herder, Meng-Day (Mandel) Yu, Farinaz Koushanfar, and Srinivas Devadas. Physical unclonable functions and applications: A tutorial. *Proceedings of the IEEE*, 102(8):1126–1141, 2014.

[142] G. Edward Suh and Srinivas Devadas. Physical unclonable functions for device authentication and secret key generation. In *DAC*, pages 9–14, 2007.

[143] Keaton Mowery, Dillon Bogenreif, Scott Yilek, and Hovav Shacham. Fingerprinting information in JavaScript implementations. In *Proceedings of W2SP*, volume 2, 2011.

[144] Elie Bursztein, Artem Malyshev, Tadek Pietraszek, and Kurt Thomas. Picasso: Lightweight device class fingerprinting for web clients. In *SPSM@CCS*, pages 93–102, 2016.

[145] Hugo Jonker, Benjamin Krumnow, and Gabry Vlot. Fingerprint surface-based detection of web bot detectors. In *ESORICS*, pages 586–605, 2019.

[146] Antoine Vastel, Walter Rudametkin, Romain Rouvoy, and Xavier Blanc. FP-Crawlers: Studying the resilience of browser fingerprinting to block crawlers. In *MADWeb*, 2020.

[147] Furkan Alaca and Paul C. van Oorschot. Device fingerprinting for augmenting web authentication: classification and analysis of methods. In *ACSAC*, pages 289–301, 2016.

[148] Pierre Laperdrix, Gildas Avoine, Benoit Baudry, and Nick Nikiforakis. Morellian analysis for browsers: Making web authentication stronger with canvas fingerprinting. In *DIMVA*, pages 43–66, 2019.

[149] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juárez, Arvind Narayanan, and Claudia Díaz. The web never forgets: Persistent tracking mechanisms in the wild. In *CCS*, pages 674–689, 2014.

[150] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. In *CCS*, pages 1388–1401, 2016.

[151] Károly Boda, Ádám Máté Földes, Gábor György Gulyás, and Sándor Imre. User tracking on the web via cross-browser fingerprinting. In *NordSec*, pages 31–46, 2011.

[152] Peter Eckersley. How unique is your web browser? In *PETS*, pages 1–18, 2010.

[153] Gabi Nakibly, Gilad Shelef, and Shiran Yudilevich. Hardware fingerprinting using HTML5. *CoRR*, abs/1503.01408, 2015.

[154] WebGL. `https://www.khronos.org/webgl/`.

[155] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints. In *IEEE SP*, pages 878–894, 2016.

[156] Jae W. Lee, Daihyun Lim, Blaise Gassend, G. Edward Suh, Marten Van Dijk, and Srini Devadas. A technique to build a secret key in integrated circuits with identification and authentication applications. In *In Proceedings of the IEEE VLSI Circuits Symposium*, pages 176–179, 2004.

[157] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.

[158] Andy Liaw and Matthew Wiener. Classification and regression by randomForest. *R news*, 2(3):18–22, 2002.

[159] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015.

[160] Samsung Remote Test Lab. `https://developer.samsung.com/remotetestlab`.

[161] Yinzhi Cao, Song Li, and Erik Wijmans. (cross-)browser fingerprinting via OS and hardware level features. In *NDSS*, 2017.

[162] FingerprintJS. `https://valve.github.io/fingerprintjs2/`.

[163] Iskander Sánchez-Rola, Igor Santos, and Davide Balzarotti. Clock around the clock: Time-based device fingerprinting. In *CCS*, pages 1502–1514, 2018.

[164] Alex Christensen. Reduce resolution of performance.now. `https://developer.mozilla.org/en-US/docs/Web/API/Performance/now`, 2015.

[165] Boris Zbarsky. Clamp the resolution of performance.now() calls to 5us. `https://hg.mozilla.org/integration/mozilla-inbound/rev/48ae8b5e62ab`, 2015.

[166] Chromium Project. window.performance.now does not support sub-millisecond precision on windows. `https://bugs.chromium.org/p/chromium/issues/detail?id=158234#c110`, 2016.

[167] Mike Perry. Bug 1517: Reduce precision of time for JavaScript. `https://gitweb.torproject.org/user/mikeperry/tor-browser.git/commit/?h=bug1517`, 2015.

[168] Pietro Frigo, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. Grand pwning unit: Accelerating microarchitectural attacks with the GPU. In *IEEE SP*, pages 195–210, 2018.

[169] Site isolation – the chromium projects. `https://www.chromium.org/Home/chromium-security/site-isolation`, 2021.

[170] Markus Moenig. Issue 820891: Webgl2: EXT_disjoint_timer_query_webgl2 failing in beta of 65. `https://bugs.chromium.org/p/chromium/issues/detail?id=820891`, 2018.

[171] Thomas Rokicki, Clémentine Maurice, and Pierre Laperdrix. SoK: In Search of Lost Time: A Review of JavaScript Timers in Browsers. In *6th IEEE European Symposium on Security and Privacy (EuroS&P'21)*, Vienna, Austria, September 2021.

[172] gpu_timing.cc – chromium code search. `https://source.chromium.org/chromium/chromium/src/+/master:ui/gl/gpu_timing.cc;l=309;drc=e5a38eddbdf45d7563a00d019debd11b803af1bb`, 2021.

[173] Brave. `https://brave.com/`.

[174] Edge. `https://www.microsoft.com/en-us/edge`.

[175] Opera. `https://www.opera.com/`.

[176] Yandex browser. `https://browser.yandex.ru/beta/`.

[177] Antoine Vastel, Pierre Laperdrix, Walter Rudametkin, and Romain Rouvoy. Fp-Scanner: The privacy implications of browser fingerprint inconsistencies. In *USENIX Security*, pages 135–150, 2018.

[178] Stjepan Picek. Challenges in deep learning-based profiled side-channel analysis. In *SPACE*, pages 9–12, 2019.

[179] scikit-optimize: Sequential model-based optimization in python. `https://scikit-optimize.github.io/stable/`.

[180] Adobe's end of life. `https://www.adobe.com/products/flashplayer/end-of-life.html`.

[181] Brave's shields. `https://support.brave.com/hc/en-us/articles/360022973471-What-is-Shields-`.

[182] Ghostery. `http:wwww.ghostery.com`.

[183] Privacy badger. `https://support.brave.com/hc/en-us/articles/360022973471-What-is-Shields-`.

[184] Imane Fouad, Nataliia Bielova, Arnaud Legout, and Natasa Sarafijanovic-Djukic. Missed by filter lists: Detecting unknown third-party trackers with invisible pixels. *PoPETs*, 2020(2):499–518, 2020.

[185] Umar Iqbal, Steven Englehardt, and Zubair Shafiq. Fingerprinting the fingerprinters: Learning to detect browser fingerprinting behaviors. In *IEEE SP*, pages 283–301, 2021.

[186] Peter Snyder, Cynthia Bagier Taylor, and Chris Kanich. Most websites don't need to vibrate: A cost-benefit approach to improving browser security. In *CCS*, 2017.

[187] Shujiang Wu, Song Li, Yinzhi Cao, and Ningfei Wang. Rendered private: Making GLSL execution uniform to prevent WebGL-based browser fingerprinting. In *USENIX Security*, pages 1645–1660, 2019.

[188] Michael Schwarz, Clémentine Maurice, Daniel Gruss, and Stefan Mangard. Fantastic timers and where to find them: High-resolution microarchitectural attacks in JavaScript. In *Financial Cryptography and Data Security*, pages 247–267, 2017.

[189] Anatoly Shusterman, Ayush Agarwal, Sioli O'Connell, Daniel Genkin, Yossi Oren, and Yuval Yarom. Prime+Probe 1, JavaScript 0: Overcoming browser-based side-channel defenses. In *USENIX Security*, 2021.

[190] Changelog from v80.0.3987.163 to v.81.0.4044.92 – chromium git repository. `https://chromium.googlesource.com/chromium/src/+log/80.0.3987.163..81.0.4044.92?pretty=fuller&n=10000`, 2020.

[191] Jóakim von Kistowski, Hansfried Block, John Beckett, Cloyce Spradling, Klaus-Dieter Lange, and Samuel Kounev. Variations in cpu power consumption. In *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering*, ICPE '16, page 147–158, New York, NY, USA, 2016. Association for Computing Machinery.

[192] Kenneth Russell. Issue 859249: Extend WebGL 2.0 compute flag expiry to M85. `https://chromium.googlesource.com/chromium/src.git/+/96186af9c385db253bf85f06f1324a729684cb2f`, 2020.

[193] Google Kenneth Russell. personal communication.

[194] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *IEEE SP*, pages 541–555, 2013.

[195] Alejandro Gómez-Boix, Pierre Laperdrix, and Benoit Baudry. Hiding in the crowd: an analysis of the effectiveness of browser fingerprinting at large scale. In *WWW*, pages 309–318, 2018.

[196] Jiexin Zhang, Alastair R. Beresford, and Ian Sheret. SensorID: Sensor calibration fingerprinting for smartphones. In *IEEE SP*, pages 638–655, 2019.

[197] Sanorita Dey, Nirupam Roy, Wenyuan Xu, Romit Roy Choudhury, and Srihari Nelakuditi. Accelprint: Imperfections of accelerometers make smartphones trackable. In *NDSS*, 2014.

[198] Anupam Das, Nikita Borisov, and Edward Chou. Every move you make: Exploring practical issues in smartphone motion sensor fingerprinting and countermeasures. *PoPETs*, 2018(1):88–108, 2018.

[199] Anupam Das, Gunes Acar, Nikita Borisov, and Amogh Pradeep. The web's sixth sense: A study of scripts accessing smartphone sensors. In *CCS*, pages 1515–1532, 2018.

[200] Hristo Bojinov, Yan Michalevsky, Gabi Nakibly, and Dan Boneh. Mobile device identification via sensor fingerprinting. *CoRR*, abs/1408.1416, 2014.

[201] W. B. Clarkson. *Breaking Assumptions: Distinguishing. Between Seemingly Identical Items Using Cheap Sensors*. PhD thesis, Princeton, 2012.

[202] Anupam Das and Nikita Borisov. Poster: Fingerprinting smartphones through speaker. In *Poster at the IEEE Security and Privacy Symposium*, 2014.

[203] Anupam Das, Nikita Borisov, and Matthew Caesar. Do you hear what I hear?: Fingerprinting smart devices through embedded acoustic components. In *CCS*, pages 441–452, 2014.

[204] Zhe Zhou, Wenrui Diao, Xiangyu Liu, and Kehuan Zhang. Acoustic fingerprinting revisited: Generate stable device ID stealthily with inaudible sound. In *CCS*, pages 429–440, 2014.

[205] Gianmarco Baldini and Irene Amerini. Smartphones identification through the built-in microphones with convolutional neural network. *IEEE Access*, 7:158685–158696, 2019.

[206] Zhongjie Ba, Sixu Piao, Xinwen Fu, Dimitrios Koutsonikolas, Aziz Mohaisen, and Kui Ren. ABC: enabling smartphone authentication with built-in camera. In *NDSS*, 2018.

[207] Davide Cozzolino and Luisa Verdoliva. Noiseprint: A CNN-based camera model fingerprint. *IEEE TIFS*, 15:144–159, 2020.

[208] Amani Al-Shawabka, Francesco Restuccia, Salvatore D'Oro, Tong Jian, Bruno Costa Rendon, Nasim Soltani, Jennifer G. Dy, Stratis Ioannidis, Kaushik R. Chowdhury, and Tommaso Melodia. Exposing the fingerprint: Dissecting the impact of the wireless channel on radio fingerprinting. In *INFOCOM*, pages 646–655, 2020.

[209] Ioannis Agadakos, Nikolaos Agadakos, Jason Polakis, and Mohamed R. Amer. Chameleons' oblivion: Complex-valued deep neural networks for protocol-agnostic RF device fingerprinting. In *EuroS&P*, pages 322–338, 2020.

[210] Nikolaos Athanasios Anagnostopoulos, Tolga Arul, Yufan Fan, Christian Hatzfeld, André Schaller, Wenjie Xiong, Manishkumar Jain, Muhammad Umair Saleem, Jan Lotichius, Sebastian Gabmeyer, Jakub Szefer, and Stefan Katzenbeisser. Intrinsic run-time row hammer PUFs: Leveraging the row hammer effect for run-time cryptography and improved security. *Cryptography*, 2(3):13, 2018.

[211] Daniel E. Holcomb, Wayne P. Burleson, and Kevin Fu. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Trans. Computers*, 58(9):1198–1210, 2009.

[212] Daniel E. Holcomb, Wayne P. Burleson, and Kevin Fu. Initial SRAM state as a fingerprint and source of true random numbers for RFID tags. In *Proceedings of the Conference on RFID Security*, volume 7, page 01, 2007.

[213] Ulrich Rührmair, Srinivas Devadas, and Farinaz Koushanfar. Security based on physical unclonability and disorder. In *Introduction to Hardware Security and Trust*, pages 65–102. Springer, 2012.

[214] Number of smartphone users from 2016 to 2021 – Statista. `https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/`, 2021.

[215] Number of mobile gaming users worldwide in 2021, by region – Statista. `https://www.statista.com/statistics/512112/number-mobile-gamers-world-by-region/`, 2021.

[216] State of Mobile 2022 – App Annie. `https://www.appannie.com/en/go/state-of-mobile-2022`, 2022.

[217] The Games Market and Beyond in 2021: The Year in Numbers – Newzoo. https://newzoo.com/insights/articles/the-games-market-in-2021-the-year-in-numbers-esports-cloud-gaming/, 2021.

[218] 75% of Pandemic-Driven Increase in Mobile Gaming Activity Will Persist Indefinitely, According to New IDC and LoopMe Report – IDC. https://www.idc.com/getdoc.jsp?containerId=prUS47906621, 2020.

[219] Sensor Tower: Mobile game spending hit $22.2B in 2021 Q1, up 25% from 2020 – VentureBeat. https://venturebeat.com/2021/04/05/sensor-tower-mobile-game-spending-hit-22-2b-in-2021-q1-up-25-from-2020/, 2021.

[220] Ailie KY Tang. Mobile app monetization: app business models in the digital era. *International Journal of Innovation, Management and Technology*, 7(5):224, 2016.

[221] Google Play vs the iOS App Store — Store Stats for Mobile Apps – 42Matters. https://42matters.com/stats, 2021.

[222] Battle Pass: Examples in Top-Grossing Games & Best Practices. https://www.blog.udonis.co/mobile-marketing/mobile-games/battle-pass, 2021.

[223] 2021 Gaming Report – Unity. https://create.unity3d.com/2021-game-report, 2021.

[224] Latest Firefox rolls out Enhanced Tracking Protection 2.0; blocking redirect trackers by default – Mozilla. https://blog.mozilla.org/en/products/firefox/latest-firefox-rolls-out-enhanced-tracking-protection-2-0-blocking-redirect-trackers-by-default/, 2021.

[225] Intelligent Tracking Protection – WebKit. https://webkit.org/blog/category/privacy/, 2021.

[226] Privacy features – Brave browser. https://brave.com/privacy-features/, 2021.

[227] The future of ads and privacy – Mozilla. https://blog.mozilla.org/en/mozilla/the-future-of-ads-and-privacy/, 2021.

[228] Building Consumer Confidence Through Transparency and Control – Cisco. https://www.cisco.com/c/dam/en_us/about/doing_business/trust-center/docs/cisco-cybersecurity-series-2021-cps.pdf, 2021.

[229] Relaxing with a Game? Enjoy an Ad – eMarketer. https://www.emarketer.com/content/mobile-gamers-more-receptive-to-ads, 2018.

[230] Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden, Alexandre Bartel, Jacques Klein, Yves Le Traon, Damien Octeau, and Patrick McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. *Acm Sigplan Notices*, 49(6):259–269, 2014.

[231] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):1–29, 2014.

[232] Clint Gibler, Jonathan Crussell, Jeremy Erickson, and Hao Chen. Androidleaks: Automatically detecting potential privacy leaks in android applica-

tions on a large scale. In *International Conference on Trust and Trustworthy Computing*, pages 291–307. Springer, 2012.

[233] Joel Reardon, Álvaro Feal, Primal Wijesekera, Amit Elazari Bar On, Narseo Vallina-Rodriguez, and Serge Egelman. 50 ways to leak your data: An exploration of apps' circumvention of the android permissions system. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 603–620, Santa Clara, CA, August 2019. USENIX Association.

[234] Martina Lindorfer, Matthias Neugschwandtner, Lukas Weichselbaum, Yanick Fratantonio, Victor Van Der Veen, and Christian Platzer. Andrubis–1,000,000 apps later: A view on current android malware behaviors. In *2014 third international workshop on building analysis datasets and gathering experience returns for security (BADGERS)*, pages 3–17. IEEE, 2014.

[235] Abbas Razaghpanah, Rishab Nithyanand, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Mark Allman, Christian Kreibich, Phillipa Gill, et al. Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem. In *The 25th Annual Network and Distributed System Security Symposium (NDSS 2018)*, 2018.

[236] Irwin Reyes, Primal Wijesekera, Joel Reardon, Amit Elazari Bar On, Abbas Razaghpanah, Narseo Vallina-Rodriguez, and Serge Egelman. "won't somebody think of the children?" examining coppa compliance at scale. *Proceedings on Privacy Enhancing Technologies*, 2018(3):63–83, 2018.

[237] Konrad Kollnig, Anastasia Shuba, Reuben Binns, Max Van Kleek, and Nigel Shadbolt. Are iphones really better for privacy? comparative study of ios and android apps, 2021.

[238] Suranga Seneviratne, Harini Kolamunna, and Aruna Seneviratne. A measurement study of tracking in paid mobile applications. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 1–6, 2015.

[239] Catherine Han, Irwin Reyes, Amit Elazari Bar On, Joel Reardon, Álvaro Feal, Serge Egelman, Narseo Vallina-Rodriguez, et al. Do you get what you pay for? comparing the privacy behaviors of free vs. paid apps. In *Workshop on Technology and Consumer Protection (ConPro 2019), in conjunction with the 39th IEEE Symposium on Security and Privacy, 23 May 2019, San Francisco, CA, USA.*, 2019.

[240] Takuya WATANABE, Mitsuaki AKIYAMA, Fumihiro KANEI, Eitaro SHIOJI, Yuta TAKATA, Bo SUN, Yuta ISHII, Toshiki SHIBAHARA, Takeshi YAGI, and Tatsuya MORI. Study on the vulnerabilities of free and paid mobile apps associated with software library. *IEICE Transactions on Information and Systems*, E103.D(2):276–291, 2020.

[241] Yuta Ishii, Takuya Watanabe, Fumihiro Kanei, Yuta Takata, Eitaro Shioji, Mitsuaki Akiyama, Takeshi Yagi, Bo Sun, and Tatsuya Mori. Understanding the security management of global third-party android marketplaces. In *Proceedings of the 2nd ACM SIGSOFT International Workshop on App Market Analytics*, pages 12–18, 2017.

[242] Nicolas Viennot, Edward Garcia, and Jason Nieh. A measurement study of google play. In *The 2014 ACM international conference on Measurement and modeling of computer systems*, pages 221–233, 2014.

[243] Kevin Allix, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. Androzoo: Collecting millions of android apps for the research community. In

Proceedings of the 13th International Conference on Mining Software Repositories, MSR '16, pages 468–471, New York, NY, USA, 2016. ACM.

[244] Introducing Google Play Pass – Google. https://play.google.com/about/play-pass/, 2021.

[245] Grow with Google Play Pass – Google. https://play.google.com/console/about/googleplaypass/, 2021.

[246] Lists of APKs – AndroZoo. https://androzoo.uni.lu/lists, 2021.

[247] Google-Play-Scraper – PyPi. https://pypi.org/project/google-play-scraper/, 2021.

[248] Exodus – The privacy audit platform for Android applications. https://reports.exodus-privacy.eu.org/, 2021.

[249] Exodus Tracker Investigation Platform. https://etip.exodus-privacy.eu.org/trackers/all, 2021.

[250] Permissions on Android – Google. https://developer.android.com/guide/topics/permissions/overview, 2021.

[251] RECEIVE_BOOT_COMPLETED permission automatically added – Google Mobile Ads SDK Developers. https://groups.google.com/g/google-admob-ads-sdk/c/2OXYkWnMmI0, 2021.

[252] [GA4] Games reports – Google. https://support.google.com/analytics/answer/9713967, 2021.

[253] Mobile App Trends Report – LiftOff. https://liftoff.io/wp-content/uploads/2019/10/2019-Mobile-App-Trends-Report.pdf, 2019.

[254] Designing Apps for Children and Families – Android Developers. https://support.google.com/googleplay/android-developer/answer/9893335, 2021.

[255] Participate in the Families Ads Program – Android Developers. https://support.google.com/googleplay/android-developer/answer/9283445, 2021.

[256] Families Ads Program – Android Developers. https://support.google.com/googleplay/android-developer/answer/9900633, 2021.

[257] User Privacy and Data Use – Apple. https://developer.apple.com/app-store/user-privacy-and-data-use/, 2021.

[258] Privacy regulations and comeback of contextual advertising – Gala. http://blog.galalaw.com/post/102h4vl/privacy-regulations-and-comeback-of-contextual-advertising, 2021.

[259] Launching Data safety in Play Console: Elevating Privacy and Security for your users – Android Developers Blog. https://android-developers.googleblog.com/2021/10/launching-data-safety-in-play-console.html, 2021.

[260] About Android App Bundles – Android Developers. https://developer.android.com/guide/app-bundle, 2021.

[261] Number of internet and social media users worldwide as of June 2023 - Statista. https://www.statista.com/statistics/617136/digital-population-worldwide/, 2023.

[262] How Much Data Do We Create Every Day? https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/, 2018.

[263] Iskander Sanchez-Rola, Xabier Ugarte-Pedrero, Igor Santos, and Pablo G Bringas. The web is watching you: A comprehensive review of web-tracking techniques and countermeasures. *Logic Journal of the IGPL*, 25(1):18–29, 2017.

[264] Nayanamana Samarasinghe and Mohammad Mannan. Towards a global perspective on web tracking. *Computers & Security*, 87:101569, 2019.

[265] Climate Change Indicators: Greenhouse Gases. https://www.epa.gov/climate-indicators/greenhouse-gases, 2022.

[266] Data Centres and Data Transmission Networks. https://www.iea.org/reports/data-centres-and-data-transmission-networks, 2021.

[267] Brian J Watson, Amip J Shah, Manish Marwah, Cullen E Bash, Ratnesh K Sharma, Christopher E Hoover, Tom W Christian, and Chandrakant D Patel. Integrated design and management of a sustainable data center. In *International Electronic Packaging Technical Conference and Exhibition*, volume 43604, pages 635–644, 2009.

[268] Atefeh Khosravi, Saurabh Kumar Garg, and Rajkumar Buyya. Energy and carbon-efficient placement of virtual machines in distributed cloud data centers. In *European Conference on Parallel Processing*, pages 317–328. Springer, 2013.

[269] Atefeh Khosravi, Lachlan LH Andrew, and Rajkumar Buyya. Dynamic vm placement method for minimizing energy and carbon cost in geographically distributed cloud data centers. *IEEE Transactions on Sustainable Computing*, 2(2):183–196, 2017.

[270] Atefeh Khosravi and Rajkumar Buyya. Energy and carbon footprint-aware management of geo-distributed cloud data centers: A taxonomy, state of the art, and future directions. *Sustainable Development: Concepts, Methodologies, Tools, and Applications*, pages 1456–1475, 2018.

[271] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. *arXiv preprint arXiv:1806.01156*, 2018.

[272] IAB. Internet Advertising Revenue Report: Full Year 2021. https://www.iab.com/insights/internet-advertising-revenue-report-full-year-2022/, 2022.

[273] Enric Pujol, Oliver Hohlfeld, and Anja Feldmann. Annoyed users: Ads and ad-block usage in the wild. In *Proceedings of the 2015 Internet Measurement Conference*, pages 93–106, 2015.

[274] Jiaping Gui, Stuart Mcilroy, Meiyappan Nagappan, and William GJ Halfond. Truth in advertising: The hidden cost of mobile ads for software developers. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 1, pages 100–110. IEEE, 2015.

[275] Statista. Adblocking penetration rate in 2021. https://www.statista.com/statistics/351862/adblocking-usage/, 2021.

[276] Joshua M Pearce. Energy conservation with open source ad blockers. *Technologies*, 8(2):18, 2020.

[277] OpenRTB. `https://www.iab.com/guidelines/openrtb/`, 2010.

[278] John Cook, Rishab Nithyanand, and Zubair Shafiq. Inferring tracker-advertiser relationships in the online advertising ecosystem using header bidding. *Proceedings on Privacy Enhancing Technologies*, 1:65–82, 2020.

[279] Codecarbon. `https://codecarbon.io/`, 2023.

[280] Maaz Bin Musa and Rishab Nithyanand. Atom: Ad-network tomography. *Proceedings on Privacy Enhancing Technologies*, 4:295–313, 2022.

[281] Muhammad Ahmad Bashir, Sajjad Arshad, William Robertson, and Christo Wilson. Tracing information flows between ad exchanges using retargeted ads. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 481–496, 2016.

[282] Lukasz Olejnik, Tran Minh-Dung, and Claude Castelluccia. Selling off privacy at auction. 2013.

[283] Syed Suleman Ahmad, Muhammad Daniyal Dar, Muhammad Fareed Zaffar, Narseo Vallina-Rodriguez, and Rishab Nithyanand. Apophanies or epiphanies? how crawlers impact our understanding of the web. In *Proceedings of The Web Conference 2020*, pages 271–280, 2020.

[284] Jordan Jueckstock, Shaown Sarker, Peter Snyder, Aidan Beggs, Panagiotis Papadopoulos, Matteo Varvello, Benjamin Livshits, and Alexandros Kapravelos. Towards realistic and reproducibleweb crawl measurements. In *Proceedings of the Web Conference 2021*, pages 80–91, 2021.

[285] uBlock Origin. `https://ublockorigin.com/`, 2022.

[286] Nikhil Jha, Martino Trevisan, Luca Vassio, and Marco Mellia. The internet with privacy policies: Measuring the web upon consent. *ACM Transactions on the Web (TWEB)*, 16(3):1–24, 2022.

[287] Waqar Aqeel, Debopam Bhattacherjee, Balakrishnan Chandrasekaran, P Brighten Godfrey, Gregory Laughlin, Bruce Maggs, and Ankit Singla. Untangling header bidding lore: Some myths, some truths, and some hope. In *Passive and Active Measurement: 21st International Conference, PAM 2020, Eugene, Oregon, USA, March 30–31, 2020, Proceedings 21*, pages 280–297. Springer, 2020.

[288] Vlad C Coroama, Lorenz M Hilty, Ernst Heiri, and Frank M Horn. The direct energy demand of internet data flows. *Journal of Industrial Ecology*, 17(5):680–688, 2013.

[289] Dan Schien and Chris Preist. Approaches to energy intensity of the internet. *IEEE Communications Magazine*, 52(11):130–137, 2014.

[290] Marion Ficher, Françoise Berthoud, Anne-Laure Ligozat, Patrick Sigonneau, Badis Tebbani, and Maxime Wisslé. Rapport: évaluation de l'empreinte carbone de la transmission d'un gigaoctet de données sur le réseau renater. Technical report, Tech. Rep., 2021.[Online]. Available: https://ecoinfo. cnrs. fr/wp-content . . . , 2021.

[291] Yanan Liu, Xiaoxia Wei, Jinyu Xiao, Zhijie Liu, Yang Xu, and Yun Tian. Energy consumption and emission mitigation prediction based on data center traffic and pue for global data centers. *Global Energy Interconnection*, 3(3):272–282, 2020.

[292] Electrictiymap. `https://electricymaps.net`, 2023.

[293] Carbon intensity per country – ourworldindata. `https://ourworldindata.org/grapher/carbon-intensity-electricity`, 2023.

[294] Symantec's webpulse site review. `https://sitereview.bluecoat.com`, 2023.

[295] Sacha Wunsch-Vincent. Online news: Recent developments, new business models and future prospects. *The Changing Business of Journalism and its Implications for Democracy*, pages 25–37, 2010.

[296] Chin Osathanunkul. A classification of business models in video game industry. *International Journal of Management Cases*, 17(1):35–44, 2015.

[297] Brian Daigle. Data centers around the world: A quick look. *United States International Trade Commission: Washington, DC, USA*, 2021.

[298] Robin Clayton. How advertisers can take the lead in reducing carbon emissions. `https://martech.org/how-advertisers-can-take-the-lead-in-reducing-carbon-emissions/`, 2023.

[299] Colin M Gray, Cristiana Santos, Nataliia Bielova, Michael Toth, and Damian Clifford. Dark patterns and the legal requirements of consent banners: An interaction criticism perspective. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–18, 2021.

[300] Anthony Chavez. Expanding testing for the privacy sandbox for the web – google blog. `https://blog.google/products/chrome/update-testing-privacy-sandbox-web/`, 2022.

[301] How to stop data centres from gobbling up the world's electricity – nature. `https://www.nature.com/articles/d41586-018-06610-y`, 2018.

[302] Donell Holloway, Lelia Green, and Sonia Livingstone. Zero to eight: Young children and their internet use. 2013.

[303] Rishabh Khandelwal, Asmit Nayak, Hamza Harkous, and Kassem Fawaz. Automated cookie notice analysis and enforcement. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 1109–1126, 2023.

[304] Ralf Gundelach and Dominik Herrmann. Cookiescanner: An automated tool for detecting and evaluating gdpr consent notices on websites. In *Proceedings of the 18th International Conference on Availability, Reliability and Security*, pages 1–8, 2023.

[305] Midas Nouwens, Rolf Bagge, Janus Bager Kristensen, and Clemens Nylandsted Klokmose. Consent-o-matic: Automatically answering consent pop-ups using adversarial interoperability. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pages 1–7, 2022.

[306] Jiyeon Lee, Hyosu Kim, and Kilho Lee. Vrkeylogger: Virtual keystroke inference attack via eavesdropping controller usage pattern in webvr. *Computers & Security*, 134:103461, 2023.

[307] Marina Botvinnik, Tomer Laor, Thomas Rokicki, Clémentine Maurice, and Yossi Oren. The finger in the power: How to fingerprint pcs by monitoring their power consumption. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 24–45. Springer, 2023.

[308] Soheil Feizi, MohammadTaghi Hajiaghayi, Keivan Rezaei, and Suho Shin. Online advertisements with llms: Opportunities and challenges. *arXiv preprint arXiv:2311.07601*, 2023.

Appendix A

Drawnapart

A.1 Deterministic attributes collected for the in-the-wild dataset

```
cookies and session support,
HTTP headers: [Accept, Accept-Encoding, Language, User-Agent],
navigator: [DNT, platform, plugins],
screen: [width, height]
timezone,
WebGL: [vendor, renderer]
```

A.2 Evaluation of the standalone pipeline on additional browsers in the wild

Table A.1: Standalone Performance of DrawnApart over multiple browsers

Browser	Accuracy (*Base rate*)		
	Top-1	**Top-5**	**Top-10**
Chrome	24.31% (*0.7%*)	49.12% (*2.9%*)	60.80% (*4.7%*)
Edge	52.60% (*2.9%*)	85.48% (*15.6%*)	93.86% (*29.7%*)
Opera	79.28% (*17.9%*)	99.41% (*50.7%*)	100.0% (*77.5%*)
Yandex	89.69% (*27.6%*)	98.36% (*85.9%*)	99.76% (*94.1%*)

Appendix B

AdCarbon

B.1 Server power measurements

Figure B.1 depicts the average power consumption of various backend Web stacks based on the number of simultaneous clients. Power consumption increases with the number of clients. However, for most Web servers, the average power consumption tends to plateau after 32 simultaneous clients.

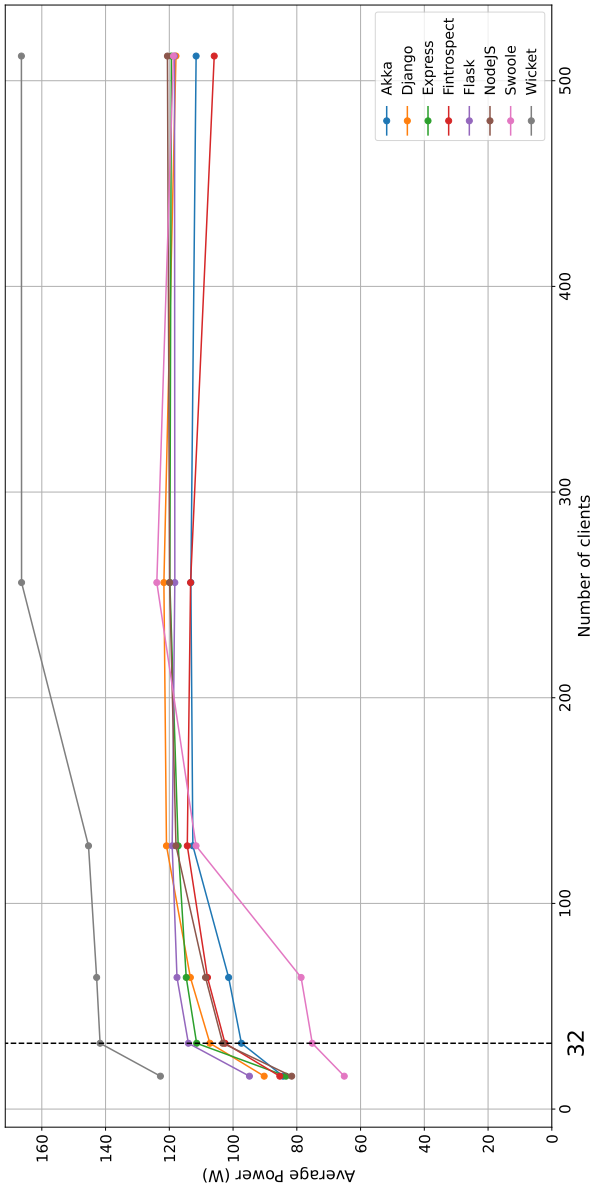


Figure B.1: Average power consumed per number of clients for each backend server