Imperial College
— of —
Science, Technology and Medicine

Faculty of Engineering
Department of Computing

MEng Computing
Individual Project Final Report

Fingerprinting the Mobile Web

Author:
John Oliver

Supervisor:
Dr. Ben Livshits

Web Version

2nd Marker:
Prof. Christopher Hankin

Presented in partial fulfilment of the requirements for the degree of Master of Engineering

2018-06-18

Abstract

Web browser fingerprinting is the extraction, usually through the use of JavaScript, of details about the computer system that that particular web browser is running on. Usually, this consists of attributes such as the browser user-agent, language, and reported time zone.

In this project, with the use of an external dataset, and a smaller and more directly curated one of our own, we come to some conclusions and comparisons about the fingerprintability of various devices, mobile and desktop, and between the two major platforms that power mobile devices today (Apple iOS and Google Android).

To this end, we create a basic fingerprinting tool to gather some of our own data to test and answer specific questions about certain devices, and we make observations about a recent but previously unpublished part of the AmIUnique.org dataset, which has been collecting data for several years as part of a different research project.

We also make observations about the effectiveness of security tools and methods, such as the use of Tracking Protection and the Tor network.

We find that fingerprinting is really quite effective on “real-world” devices, mobile and desktop, and that technology that blocks fingerprinting is often ineffective, or easily worked around by good web design. However, some fingerprint detection and protection in some browsers is good and could provide a good base to work with in improving privacy and protection from trackers and fingerprinting in other web browsers.

Acknowledgements

I'd like to thank:

Benoit Baudry of the KTH Royal Institute of Technology, Stockholm; for allowing me to use the AmIUnique.org dataset, analysis of which makes up a substantial part of this project.

Chris Hankin for agreeing to help assess this project and giving invaluable advice as I was starting out.

My parents for their unconditional support whilst I've been away from home, and for the world in general.

Maria Valera-Espina for the support that she's given me as senior tutor.

Lloyd Kamara, Geoff Bruce and Duncan White of DoC's Computing Support Group, who have variously entertained my requests for devices amongst other things over the past few years.

Imperial College Dramatic Society (and its various members and alumni), who've provided me with some fantastic opportunities over the past four years — you will be missed.

And of course, **Ben Livshits** for his advice surrounding academic work, and his domain expertise, as well as his patience in working with me. I am grateful to him for his invaluable help in this work.

"Privacy is not an option, and it shouldn't be the price we accept for just getting on the Internet."

Gary Kovacs

CEO of Mozilla Corporation 2010-13 [1]

Contents

Chapter 1

Introduction

1.1 Motivation

It is often noted by people that as they traverse the web, especially if they look at a product on a website, such as Amazon.co.uk, that they will spend the rest of the day being followed by ads for the project from site-to-site. This is known as tracking, and it uses cookies and other common web technologies to "follow" users from site-to-site, taking note of what they are looking at, and serving relevant advertisements, or rather, advertisements designed to specifically appeal to that person in that moment. More follows on this in section 2.1, focusing on the business of advertising, and section 2.1.1 with regards to the blocking of these advertisements. We'll also look at the privacy implications in section 2.1.2.

Extending from this is the idea of "fingerprinting", which is like tracking, but attempts to individually identify a device from its first use, even if cookies and other local storage are deleted (by 'local storage', we here refer to web technologies that provide local device storage for websites, as opposed to all forms of local storage). It uses a variety of metrics to find out information about your browser, such as installed fonts, screen resolution, the user-agent string sent by the browser and others. There are a number of websites that allow you to test your own browser in this respect, such as the EFF's[1] Panopticlick [2] project, and the "Am I Unique" website and research project [3]. We are often surprised by how identifiable my various browsers and devices are even on these

[1]Electronic Frontier Foundation

relatively small datasets (compared to, say, all of Google AdWords).

Fingerprinting can use novel features in new and emerging web technologies too; these include analysing WebGL renders made by browsers, analysing audio output [4], and on mobile devices, the use of hardware features such as accelerometers and compasses [5] used to identify individuals. We hope to look at this fingerprinting on mobile devices, especially iOS devices, and see how identifiable one device can be from another to a web site, even with apparently identical configurations, software versions and models of a device. This is made especially interesting on iOS with the nature of all web browsers being essentially wrappers around the built-in Apple Safari browser, and makes it harder to distinguish users. We'll take a look at current developments in section 2.2. We'll look more clearly at this in Chapter 3, and at the research around both it and online advertising and tracking in Chapter 4.

1.2 Project Objectives

We have three main objectives in this project:

- We hope to attempt to identify novel ways of fingerprinting web browsers on mobile devices, with a focus on iOS devices (which are very similar within each model). We hope that this provides a base for further research in this area, and that it will perhaps give users of mobile devices a greater degree of privacy in their online interactions.
- We also hope to investigate the current state of web browser fingerprinting technologies, and how they apply to mobile devices, especially iOS.
- We hope to build on previous research in fingerprinting iOS devices such as Kurtz et al. [6], but with a web-based approach rather than the described native mobile app.

In Chapter 7 we'll evaluate how well we've achieved these objectives.

1.3 Research Questions

We'd like to achieve this by asking the following questions:

- Which, of mobile and desktop devices, is more fingerprintable?
- On mobile devices, of the two major platforms Android and iOS, which is more fingerprintable?
- What fingerprinting vectors differentiate between identical phone models?
- What fingerprinting vectors on mobile differ from desktop devices?
- Is it easier to evade fingerprinting on mobile or desktop devices?

We'll look at answering these questions in detail through our own data collection and analysis, and analysis of an external dataset in Chapter 3, and we'll also find more context on these in Chapter 4.

1.4 Report Layout

In this report, our primary results and insights are presented in Chapter 3, and we will present the results in the format of some data and discussion, followed by the main point that we'd like to make about the data and the discussion, indicated by the phrase **Key Takeaway**.

Chapter 2

Background

In this section, we will be looking at the current state-of-the-art regarding online advertising and tracking. We'll investigate how online advertising works, the privacy implications, and what can be done to block this. We'll also look at some current fingerprinting techniques and technologies.

2.1 Business of Advertising

Web advertising is a multi-billion dollar business; according to eMarketers [7], projected spending in 2017 is to be US$ 229.25bn, rising further to US$ 335.48bn by the end of the decade. Large players in this space include Google (via its AdWords product), Facebook, and Twitter (the latter two both using social media within their own websites and embedded around the web). These are the primary revenue sources for both Facebook and Google.

Web advertising traditionally (from the early 1990's) entailed individual webmasters adding adverts to their sites. This is very rare now, though, with the vast majority of publishers of web content who run adverts choosing to effectively outsource the selection and display of advertisements to external companies, such as Google.

With any particular advertisement agency, that agency is able to track a user's presence across the internet, provided that the advertisement agency has been employed by pages across the web. This allows an ad agency to build up a profile of a user across the internet by taking note of what web pages they have been looking at — an agency can keep track of an individual through the use of cookies. The reasoning behind this is that adverts are now much more relevant to a user, making them more likely to click on them.

For example, it is much cheaper for an advertiser to pay only for adverts for some event to be shown to Women in London, who enjoy swimming, say, a women-only swimming class. There is little point paying for a man aged 70 to see such an advertisement, as they are unlikely to be able to partake in such an activity. A better-informed ad agency can therefore generate more clicks on some advert, targeted at some demographic(s), and thus charge more for their services, and make a greater profit. The advertiser also likes this because it reduces waste spent on showing adverts to people who aren't interested, thus meaning their allocation of budget is spent more directly on the demographic(s) they would actually like to advertise to. So too do publishers, who are paid by the ad agency for displaying the ads, and would also much prefer them to be relevant to their visitors. In section 2.1.2, we discuss the privacy implications of this.

Many publishers are now dependent upon advertising to survive, as noted by the FCC [8] (though some implement other revenue generators, such as a paywall, or a subsection of content is "premium', shown alongside free content[1]) [9], but in a world of increasing pressure on news publications and other outlets, the pressure to have ever-increasing numbers of adverts on a web-page has sometimes resulted in pages where it is difficult to discern advertisements from the actual page content, as well as intrusive "pop-up" adverts, either through the use of an additional browser window, or via "overlay" elements, which cover up the viewport entirely or substantially, and must be manually dismissed. This sort of intrusive advertising has possibly been the major driving force behind the beginnings of many popular ad-blockers - we'll discuss this in Section 2.1.1.

[1]This model of paid-for premium content presented alongside free, ad-supported content is used by The Telegraph, among others

2.1.1 Ad-blocking and ad-blockers

Ad-blocking is the techniques used by ad-blockers to block not only online advertisements from display to the user, but also often various tracking scripts (see section 2.1.2 for more on that). The most popular ad-blocker in current use is AdBlock Plus, first released in 2006, and now with well over 30 million users [6]. Ad-blockers use a variety of different rules to blacklist and whitelist advertisements of various types, such rules are detailed in Walls [11]. A number of studies have been carried out on various ad-blocking programs, and the rule lists that they use, [12, 13] and have found that some ad-blockers are very efficient and quick, and others not so much. Indeed, according to [12], the μBlock tool gives the best performance out of a number of mainstream ad-blockers.

An important thing to mention here is the *Acceptable Ads* program [14], previously operated by Eyeo GmbH - itself the company behind AdBlock Plus, and as of 2017 by the independent “Acceptable Ads committee‘, comprising user groups, advertising companies, and industry and academic experts [15]. Acceptable Ads defines criteria for adverts that should be permitted through Ad-blockers in order to support publishers who require advertisements to fund their work. The Acceptable Ads criteria provides a set of rules that publishers must follow for inclusion in the program, which sees the ads that follow the criteria whitelisted (by default[2]) on AdBlock Plus. Most advertising agencies do not have to pay for access, though “large” ones may be required to pay a fee to Eyeo, who use it to support their operations including the further development of AdBlock Plus and other products.

In Walls et al. [11], we see some interesting results from an analysis of the program, revealing that, at the time of its writing (Oct 2015), it allows some ads to be displayed on 59% of the top 5000 websites, plus another 2.6 million parked domains. There were some concerns about the openness of the program mentioned by Walls et al., namely that it did not seem to be clear what qualified as

[2]An individual user is at liberty to disable the *Acceptable Ads* program on their own device, and there is no restriction on doing so.

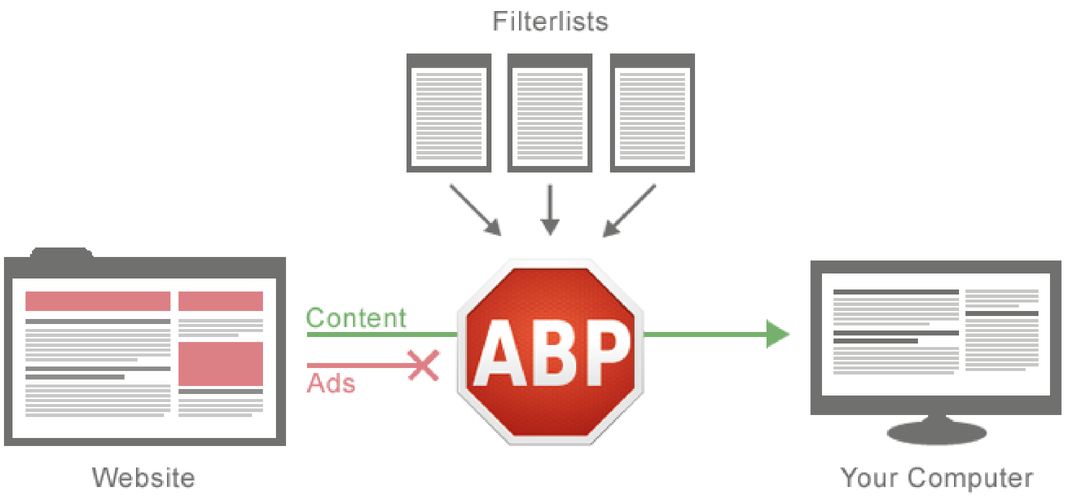


Figure 1: How AdBlock Plus Works
Source: [10]

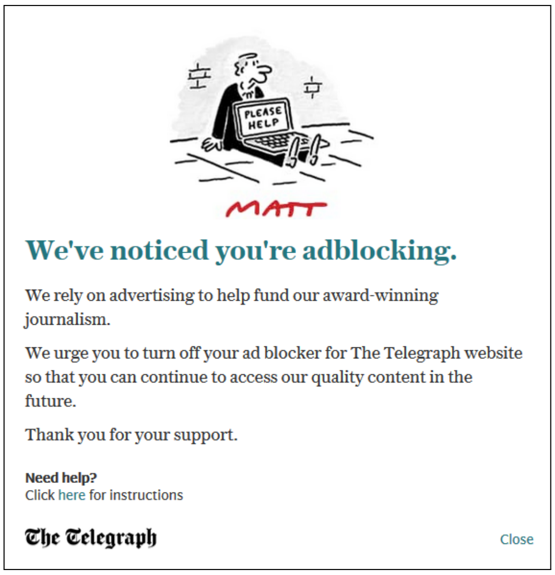
a "large" organisation - with the difference seemingly being arbitrary, and some companies that paid for access having rules added with seemingly no community input. Whether or not processes have changed in the intervening two years it yet to be seen. However, the new "Acceptable Ads Committee" will probably address these concerns head-on. It is noted that the AdBlock Plus website also reveals that there is now a formal definition of what a "large" advertiser entails , namely that it is such "when it gains more than 10 million additional ad impressions per month due to participation in the Acceptable Ads initiative' [10].

This may give the impression that all is well and good for users, but more recently [8, 16] small JavaScript programs have been developed to try and detect the use of ad-blocking software by users. As a result, some sites may refuse to display content (*forbes.com*), display a pop-up message asking the user to disable their ad-blocker (*telegraph.co.uk*), or replace adverts with a similar message about disabling the ad-blocker. These usually operate by creating a "bait" element and then testing to see if it has been deleted or otherwise hidden from view (i.e. `display=none`, `maxHeight=0`, etc.). The test may be as soon as the document loads, or a short delay after the document has loaded (to give additional time for the ad-blocking program to run), or may run every few seconds once a page has loaded (to get around any ad-blocker that may run repeatedly or after a longer delay). Developments in this area continue, and is widely expected to continue being an arms race, with one side bettering the other for a few months, and then their roles reversing [16, 17].

2.1.2 Privacy & Tracking

As mentioned in section 2.1, advertising agencies, such as Google, gain a large amount of information on users who visit pages that contain adverts served by that agency. Such information brings with it privacy concerns, because it may be used to infer demographic and personal information about individual users. The International Association of Privacy Professionals gives a hypothetical example whereby the increase of tracking users cross-devices creates the risk that some potentially compromising personal information could be mistakenly revealed to people in the vicinity, on the same network, or to advertisers. [18] Indeed, Adrian Chen, writing in *Gawker*, revealed an instance in 2010 of the latter taking place at Facebook. Stanford's Aleksandra Korolova revealed that she was able to use advertising campaign tools within Facebook to expose the "Interested in" setting of a particular user's account, which displays sexual orientation. This user had their "Interested in" privacy setting set to be visible to "Friends only", but using inference from ad impressions, was able to show that a user had the setting set one way rather than another. Facebook, upon being notified, modified their algorithms, but as Korolova notes, their apparent implementation is far from perfect [19, 20].

These sorts of issues are deeply concerning to many people who value their own privacy, and don't want necessarily for specific aspects of their self to be either discovered, or to otherwise become known through inference as described by Korolova [20]. As a result the AdChoices program was established as a self regulatory program [9, 21], along with a number of others, in order to try to dissuade consumers from



We've noticed you're adblocking.

We rely on advertising to help fund our award-winning journalism.

We urge you to turn off your ad blocker for The Telegraph website so that you can continue to access our quality content in the future.

Thank you for your support.

Need help?
Click here for instructions

The Telegraph Close

Figure 2: Anti-adblock message on *telegraph.co.uk*

Figure 3: AdChoices Icon
Source: [21]

enabling ad-blocking software (which hurts advertisers and publishers), and to try to undo the bad name and reputation that online advertising had become famous for. However, according to Vratonjic et al. [9], consumer awareness remains low and is rising only very slowly (from 5% in 2011 to 6% in 2013). The program is supposed to allow consumers to "opt-out" of being tracked on a per-browser basis (somewhat ironically, it does this by setting cookies). It is not the only technology that exists to attempt to stop tracking (or more accurately, provide a means to inform a company electronically when connecting to their servers that you do not wish to be tracked), however.

In 2011, Mozilla announced and submitted to the IETF a proposal for a new HTTP header to be sent by browsers, known as Do Not Track [22]. This allows a browser to be configured to send with all HTTP requests an "opt-out" from tracking, an explicit "opt-in', or (by default) give no preference. It is implemented as a single bit (1 or 0), and when the default “no preference” is selected, the header is not sent with requests. However, *The Register*’s Iain Thomson [4], notes that if more than 15% of users were to enable such a feature, most networks would ignore it and track them anyway. Indeed, the feature was described by Sophos as “the privacy standard that’s melting away’ [23]. Nonetheless, the feature is supported by the five major browsers (Chrome, Firefox, Internet Explorer, Opera, and Safari).

Many users also use Ad-Blockers (such as AdBlock Plus and μBlock, now renamed uBlock Origin[3]), as well as similar tools (such as Ghostery) in order to try and improve their own privacy, but as concluded by Gervais et al. [25], not all of these tools are brilliant at achieving greater privacy. Indeed, Ghostery in its default state will do nothing - one must explicitly block trackers. AdBlock Plus in its default settings will provide some protection, but to avoid tracking altogether one must change the settings of both to be their strictest in terms of blocking. For most users using adblockers, they are unlikely to have done so, because most adblockers will do an acceptable amount of adblocking with no configuration change from the defaults, and that is likely to block ads sufficiently to please the vast majority of adblocking usrs.

[3]There is a dispute over the name, however, with details available at the project’s homepage [24]

2.2 Fingerprinting

Moving away from “tracking” specifically, which usually uses cookies to determine whether or not a user is the same across websites and, in some operating systems (iOS, Windows 10), across devices, we also have the concept of “fingerprinting’, which is the attempt to extract as much information out of a device such that we can individually identify it without it necessarily providing any personal identification of the user of said device. AmIUnique.org [3] and Panopticlick [2] are good examples of websites that fingerprint people, and are able to detect a range of facts about web browsers, which when combined produce an often unique fingerprint. This is achieved with information from a number of sources including:

- Data sent to the server directly

by the web browser (e.g. IP address, User-agent - Operating System and Browser Version)

- WebGL information (Graphics Card / Driver)
- Information inferable with JavaScript (e.g. screen resolution)
- Information provided by Adobe Flash Player if installed (e.g. list of installed fonts)

Altogether these provide a remarkable fingerprint of any particular browser. For illustration, Table 1 shows a subset of information provided by one browser we have used (some details redacted & truncated):

Just from this subset of information, we know that we're using Firefox 58.0, running on Windows 7, that Adobe Flash version 28.0 is enabled, as well as information about the monitor. Most useful is probably the Fonts list, as some fonts come with specific programs; it might be possible to infer, for example, that Adobe CC products are installed, for example, if lots of fonts associated with it are present. A huge number of systems do have unusual font configurations, so the use of Flash to detect this is a very good step on the way to fingerprinting someone.

On Mobile devices we have a harder time getting the list of fonts - due to the fact that most mobile devices do not run Adobe Flash Player. This will also change from 2020 too, as Adobe drops support for the aging product - from then we can expect that most desktop browsers will no longer have the plugin installed by default. Additionally, in modern browsers, the plugin is usually available only in “Click-to-Run” mode, meaning that users must explicitly opt-in to running Flash.

When we look at our project, which is using web browsers to fingerprint mobile devices, this contrasts nicely with Kurtz et al. [6], in which we discover ways of fingerprinting iOS using a native application, which has direct access to the various iOS frameworks and certain parts of the local filesystem. Web browser access is considerably more restrictive, so doing some detailed study of techniques usable from the browser will be fascinating.

More directly related to our project is Laperdrix et al. [26], the team

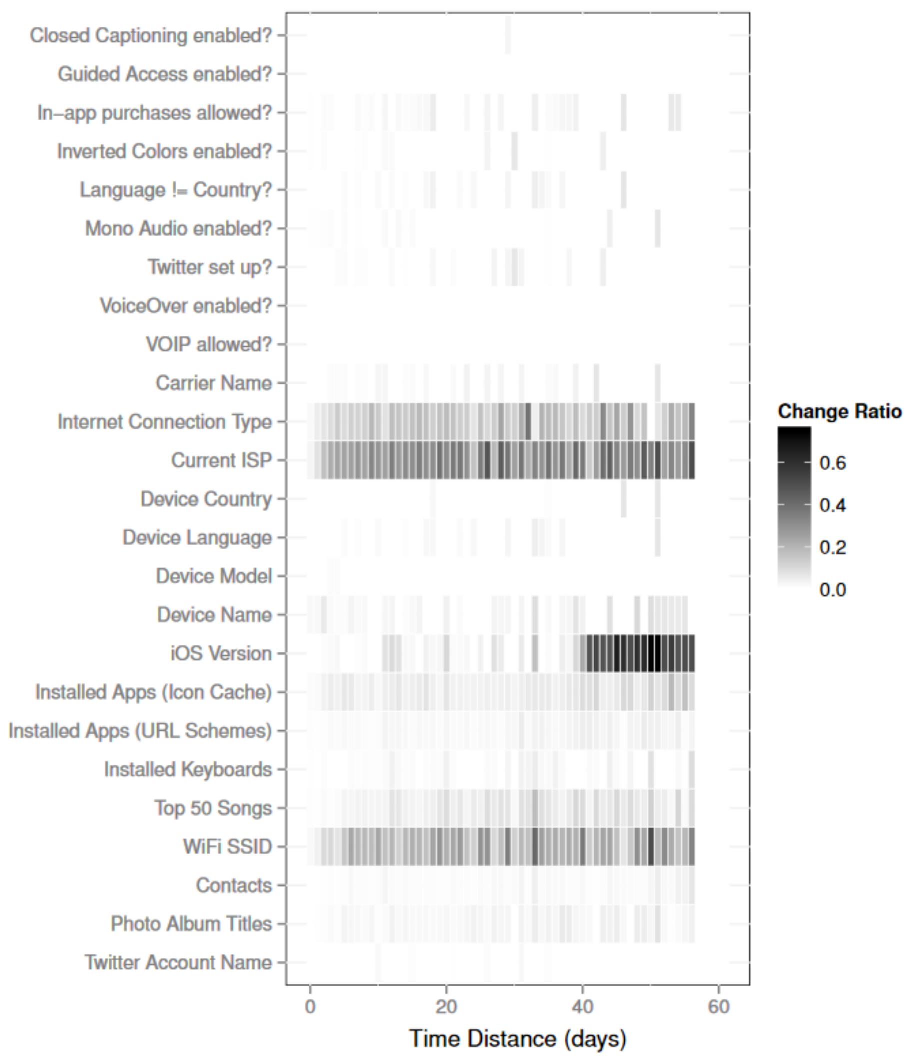


Figure 4: Fingerprintable features of iOS, and how they change over time
Source: Kurtz et al., [6] p.12.

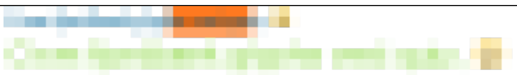
Attribute	Value
User-agent	Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:58.0) ..
Plugins	Plugin 0: Shockwave Flash; Shockwave Flash 28.0 r0..
Platform	Win64
Cookies	Enabled
Do Not Track	NC (Not sent)
Time zone	0
Resolution	1280x1024x24
HTML5 Canvas (obfuscated)	
Fonts	Arial, Arial Black, Arial Rounded MT Bold ..

Table 1: Some browser attributes used by amiunique.org to fingerprint browsers.

behind AmIUnique.org [3], which is the most recent large-scale study of fingerprinting with published results. It's not dissimilar to Panopticlick [2, 27], but we will in Chapter 3 be looking at a more recent subset of the data collected as part of the AmIUnique project.

2.3 Defeating Trackers

Tracking engines are of course extremely keen to avoid you defeating them, and a litany of browsers and browser extensions are available to that end, as well as, for the paranoid, defeating fingerprinting techniques as well.

Such solutions range from standard ad-blocking tools that we've previously described, right up to the "nuclear option" (so to speak), of running the Tor browser.

Extensions like AdBlockPlus [10], and uBlock [24] serve a very specific purpose, which is to block advertisements, though a user can certainly configure them to block more than just adverts. In these extensions, this is usually through the use of element and URL-based filters, though some other methods have been proposed. [28]

The Tor project [30] has spearheaded the development of the Tor Network, which is made up of a series of nodes, split into "relay" and "exit" nodes.

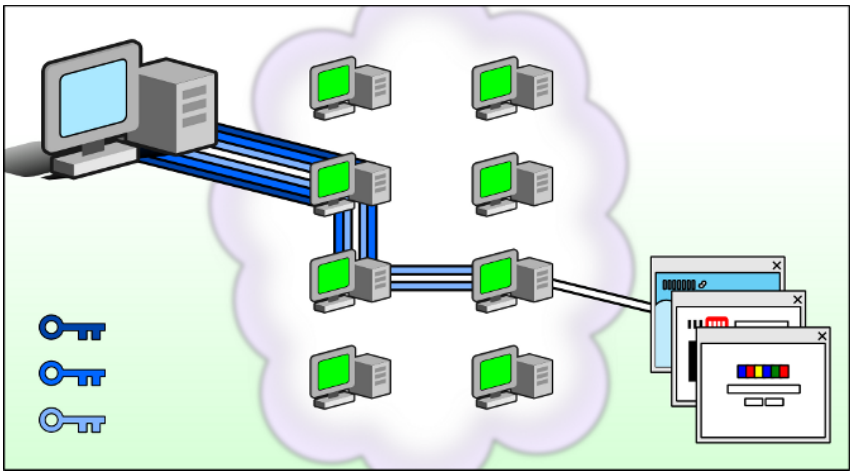
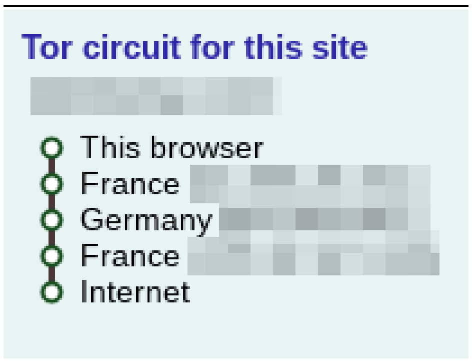


Figure 5: Routing traffic through Tor
Source: [29]

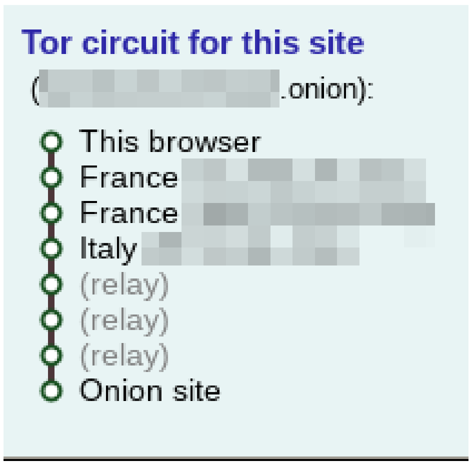
Traffic is routed at random from the client, through three separate nodes, and then onto the server. The third of the nodes is known as the "exit" node for a particular "circuit" (read connection). Each connection through a node is via an encrypted tunnel, with three layers of encrypted tunnels to the first node, then the inner two layers to the second, and then the innermost to the exit node, with no *Tor-provided* encryption between the exit node and the server. This does not preclude the use of TLS to secure websites. Figure 5 shows this visually.

As long as enough people volunteer network bandwidth, then it should be impossible to effectively spy on the network, as the only way to do so would be to control all three nodes that a particular user was using, which is unlikely as nodes are selected for use at random. As well as encryption, Tor and the Tor Browser use various other obfuscation technologies to defeat trackers and network snoopers, such as the padding of packets, and is deliberately obtuse to use plugins with. The latter is particularly noteworthy as Flash cannot be guaranteed to connect through the Tor network, thus defeating its purpose. Flash can also be used to fingerprint a desktop system much more than JavaScript alone, as it can provide the full list of installed fonts to a fingerprinting system.

Tor also provides for the creation and operation of *hidden services*, which provide a similar level of protection to website operators. This is known as the *dark web* in the media, and is used both by illegal sites, such as the infamous Silk Road website [31], and by dissident groups in countries with fewer freedoms for individuals. Some publishers and bloggers may operate exclusively via hidden services in this way, in order to protect their identity more thoroughly. Connecting to a hidden service (which have urls ending in `.onion`) effectively makes a Tor circuit with six nodes (see Figure 6), to protect both the hidden service and the user/client from being identified. The Tor browser displays this, but the identity of the three nodes that the hidden service uses are hidden from the user.


Tor circuit for this site
[redacted]
This browser
France [redacted]
Germany [redacted]
France [redacted]
Internet

(a) For ordinary websites


Tor circuit for this site
([redacted].onion):
This browser
France [redacted]
France [redacted]
Italy [redacted]
(relay)
(relay)
(relay)
Onion site

(b) For `.onion` hidden services

Figure 6: Tor Browser displays Tor circuits

Finally, Tor has a concept called *bridges*, which are like relay nodes in the Tor network, but unlike "normal" relay nodes, these are not listed in any public place. They are designed to allow access to the Tor network for those whose networks do not permit access to the Tor network directly through the blacklisting of Tor nodes (such as those in mainland China, for example). Tor also provides some protection against fingerprinting, and we'll discuss how Tor, amongst others, evades fingerprinting in Section 3.2.5.

Chapter 3

Experiments and Studies in Fingerprinting

In this section, we'll look at the way we collected data, and how we analysed external data.
We will also attempt to answer the following research questions:

1. Mobile vs. Desktop: Which device type is more fingerprintable?

2. iOS vs. Android: Which platform is more fingerprintable?

3. What sources of fingerprinting exist on identical device models?

4. How does mobile differ from desktop in fingerprinting vectors?

5. Is it easier to evade fingerprinting on mobile or desktop?

3.1 Experimental Setup

We collected both a small amount of data manually, and used a temporal slice of the AmIUnique.org dataset, which was kindly provided to me by one of the project leaders; Professor Benoit Baudry, now of the KTH Royal Institute of Technology in Stockholm.

The reason for this is that we knew it would be an enormous challenge to collect a huge amount of data, as it would require us to not only build a tool and provide sufficient resources to power potential interest from around the world, but also to publicise it as well. Typically this would be done by a research group (as is the case for both AmIUnique, and the EFF's Panopticlick project.

Such resources were not available to us, and so instead the manually-collected dataset is a much smaller and more direct look at some devices we were able to acquire.

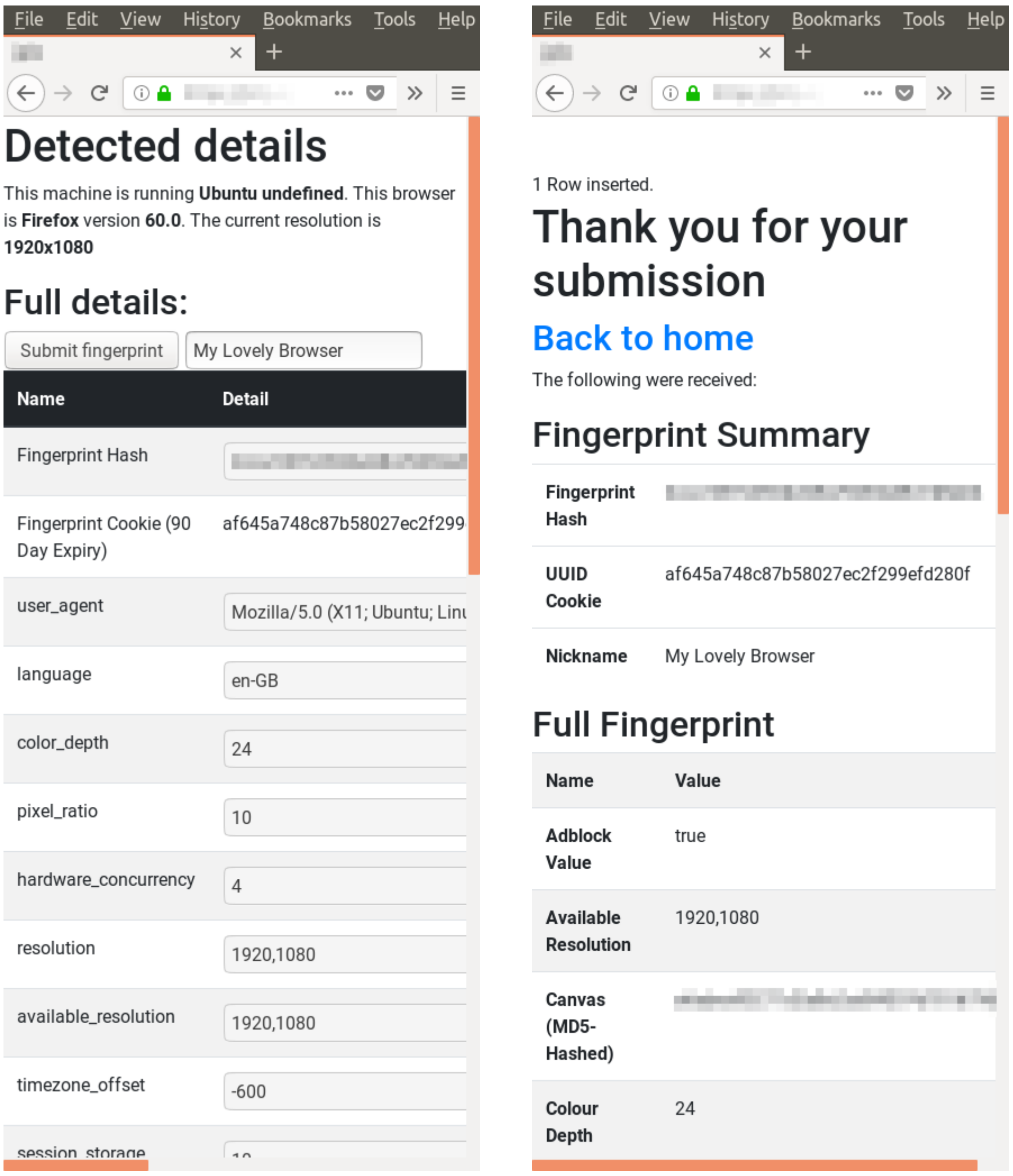
We were able to overcome this problem of scale by making use of analysis of the much larger and more complete AmIUnique.org dataset, and a more recent version than the original paper [26] had available at the time; it being published in May 2016, and the data provided to us being the three months from November 2017 to the end of January 2018. This allows us to look at some more recent developments and more current models of mobile devices that had been made available in the interim twenty-one months.

3.1.1 Manually Curated Dataset

To gather our own data on iOS, we acquired some iPads from the Imperial College Department of Computing, all the same model. We then produced a small PHP script to collect fingerprint data with JavaScript, and store it in an SQL database (see Figure 7 — it won't win any design awards, but it is functional). This method did not gain much data (nor was it meant to), but it was useful for small-scale analysis for this project. Some useful observations can be made regarding anonymity sets on iPads and some other mobile and desktop devices and browsers (see Section 3.2). This used the fingerprintjs2 library, which is an open-source JavaScript library available on GitHub [32]. The "fingerprint hash" field was generated in the browser, rather than configuring SQL to do it (see Section 3.1.2). It also uses ClientJS [33] to generate some of the niceties around specific OS and Browser names, rather than just showing the user-agent — though we don't actually store this data. We refer to the data collected as the *manually curated* dataset.

3.1.2 AmIUnique Dataset

Then some analysis was performed on the much larger (~2.6 GiB) dataset from AmIUnique.org [3]. We first had to define what variables that the original team had collected we wanted to count towards a device fingerprint, and then we had to



Detected details

This machine is running **Ubuntu undefined**. This browser is **Firefox** version **60.0**. The current resolution is **1920x1080**

Full details:

Submit fingerprint | My Lovely Browser

Name	Detail
Fingerprint Hash	[redacted]
Fingerprint Cookie (90 Day Expiry)	af645a748c87b58027ec2f299
user_agent	Mozilla/5.0 (X11; Ubuntu; Linu
language	en-GB
color_depth	24
pixel_ratio	10
hardware_concurrency	4
resolution	1920,1080
available_resolution	1920,1080
timezone_offset	-600
session_storage	10

1 Row inserted.

Thank you for your submission

Back to home

The following were received:

Fingerprint Summary

Fingerprint Hash	[redacted]
UUID Cookie	af645a748c87b58027ec2f299efd280f
Nickname	My Lovely Browser

Full Fingerprint

Name	Value
Adblock Value	true
Available Resolution	1920,1080
Canvas (MD5-Hashed)	[redacted]
Colour Depth	24

(a) The home screen (b) The submission screen

Figure 7: The tool for manual data collection

create a virtual column in our SQL database that would create an appropriate fingerprint. We did this by concatenating the chosen fields together as strings, and then creating the MD5 hash of this new, long string with all the variables contained within it — the SQL code for this is in Figure 8.

You will notice that some fields have been omitted; namely `hostHttp`, `orderHttp`, `addressHttp` and `time`.

These refer to the way the data was generated (i.e. amiunique), how the HTTP connection should be kept (usualy "close", indicating that it is closed as soon as the request is fulfilled), a hash of the IP address (which may often change on the same device, especially mobile), and the time of submission, which will also change for the same device. The rest should however remain approximately the same, unless the user consciously makes configuration changes (except for user-agent which may change if the OS or Browser are updated). We refer to this dataset as the *AIU Dataset*.

3.2 Research Questions

In this section we address a set of research questions, designed to highlight the differences in fingerprinting between mobile and desktop devices.

3.2.1 RQ1: Mobile vs. Desktop: Which is More Fingerprintable?

We would like to find out how we can compare the fingerprintability of web browsers between mobile and desktop browsers. Essentially, we want to know whether being on mobile makes one *more* or *less* identifiable?

AIU Dataset: Here, we have defined a desktop and a mobile device to be as shown by the SQL code fragments in Figure 9. The data in Figure 10, taken from the AmIUnique dataset, shows a comparison between Desktop and Mobile/Tablet devices. This clearly shows that the plurality of devices that are out there is not invisible to the browser (and therefore to websites that are visited). On Desktop devices, we are able to use WebGL to ascertain the

```sql
ALTER TABLE ``fpData`
  CHANGE ``fp_Hash` ``fp_Hash` VARCHAR(32) AS
    (md5(concat(`userAgentHttp`,`acceptHttp`,
    `connectionHttp`,`encodingHttp`,
    `languageHttp`,`pluginsJS`,`platformJS`,
    `cookiesJS`,`dntJS`,`timezoneJS`,
    `resolutionJS`,`localJS`,`sessionJS`,
    `IEDataJS`,`canvasJS`,`webGLJs`,
    `fontsFlash`,`resolutionFlash`,
    `languageFlash`,`platformFlash`,
    `adblock`,`vendorWebGLJS`,`rendererWebGLJS`,
    `octaneScore`,`sunspiderTime`,
    `pluginsJSHashed`,`canvasJSHashed`,
    `webGLJsHashed`,`fontsFlashHashed`))) VIRTUAL;
```

Figure 8: SQL generation of fingerprint hashes

```sql
-- DESKTOP OPERATING SYSTEMS
WHERE userAgentHttp LIKE '%(Windows NT%'' OR userAgentHttp LIKE
    '%(Macintosh%'' OR userAgentHttp LIKE '%X11; Linux%'

-- MOBILE OPERATING SYSTEMS
WHERE userAgentHttp LIKE '%iPad%'' OR userAgentHttp LIKE '%CPU
   iPhone OS%'' OR userAgentHttp LIKE '%Android%'
```

Figure 9: Definitions of Mobile and Desktop devices

name of the Graphics card, and from that we can infer specific information about what a desktop computer is running with regards to CPU (in the case of integrated graphics) or GPU when a discrete graphics controller is installed.

Key Takeaway: No difference in terms of unique fingerprints. The anonymity sets of size 1 correspond to 91% of all desktop entries, and 90% of all mobile (that is, Android and iOS) entries — this is shown clearly in Figure 11, which shows the top quartile of anonymity sets. We have elected to exclude other mobile platforms as they do not make up a significant enough chunk of the data, and we are most interested in common devices, the vast majority of

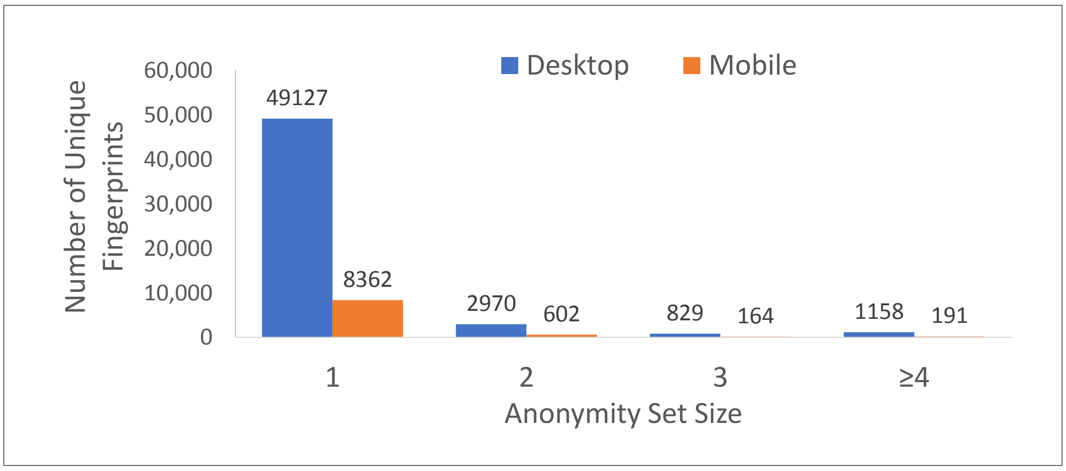


Figure 10: Desktop and Mobile/Tablet Anonymity set sizes.

which run Android or iOS.

Some other interesting information about comparing these two sets are to look at just how large the anonymity sets can be: on desktop the largest anonymity set is of size 1,729 — representing (alone) around 2% of all desktop devices. However, on Mobile, this is reduced to merely 32 — see Table 2

If we choose to interpret the data on a per-device basis rather than per-anonymity set, then the data will be weighted more towards the set sizes that balance large membership with size alone, though devices in a set of size 1 still contribute by far the largest number of members. We can see this by comparing this weighted per-device data (Figure 12) with Figure 10.

Our opinion on this question, having inspected these results, is that there is no substantial difference in anonymity granted by Desktop and Mobile browsers — at least in normal configurations — whilst the Tor Browser, for example, is almost certainly among the browsers that will have connected, the number will be insignificant compared to ordinary configurations.

It is however, important to note that desktop gets a greater number of large anonymity sets, which mobile simply cannot compete with. This allows us to tip the balance very slightly in favour of desktops for being less fingerprintable, but the vast majority of gathered fingerprints

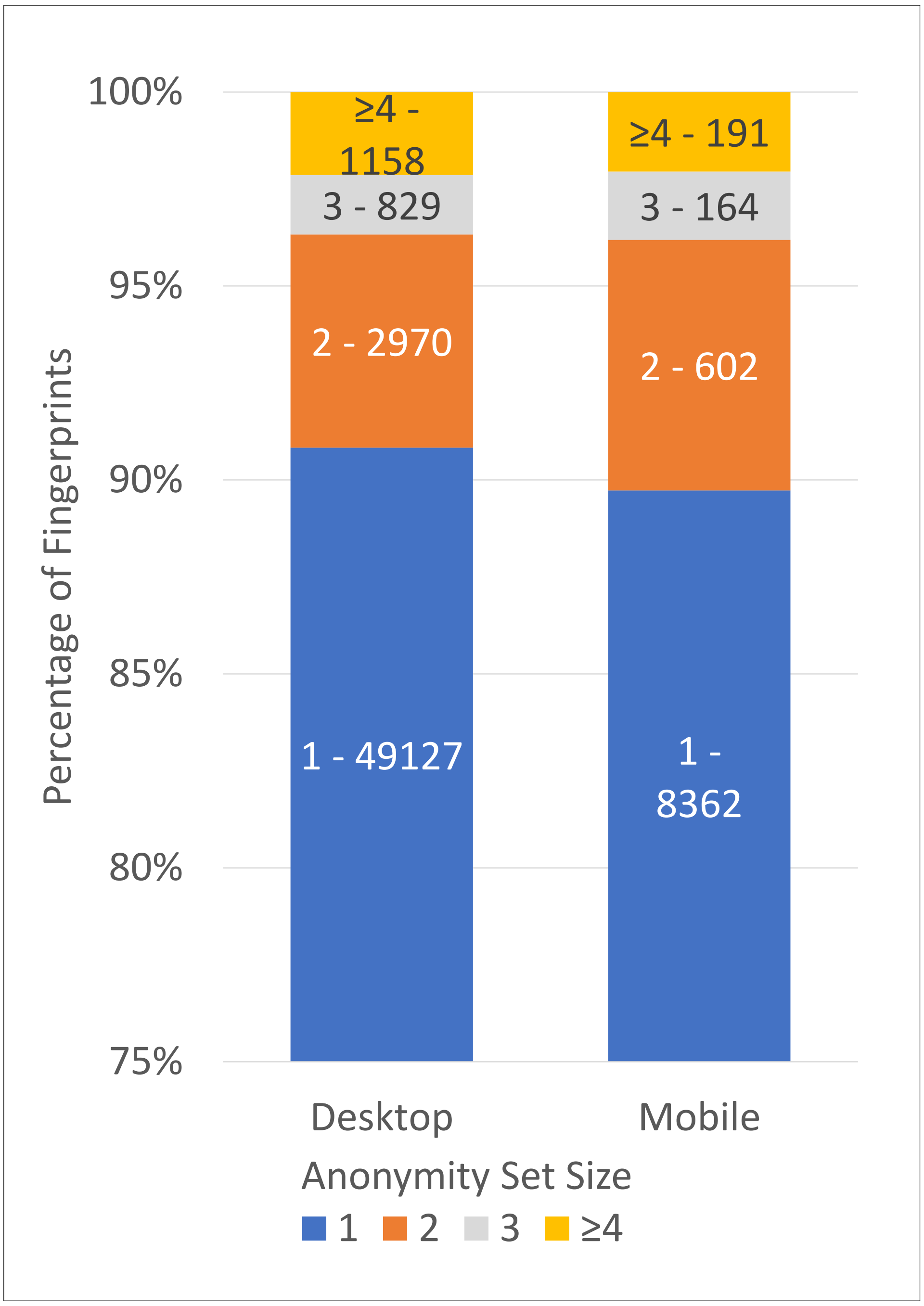


Figure 11: Desktop and Mobile anonymity sets sizes count as percentages of all anonymity sets

#	Desktop	Mobile
5	188	26
4	232	28
3	302	30
2	452	31
1	1729	32

Table 2: Top 5 largest anonymity sets for Desktop and Mobile Browsers

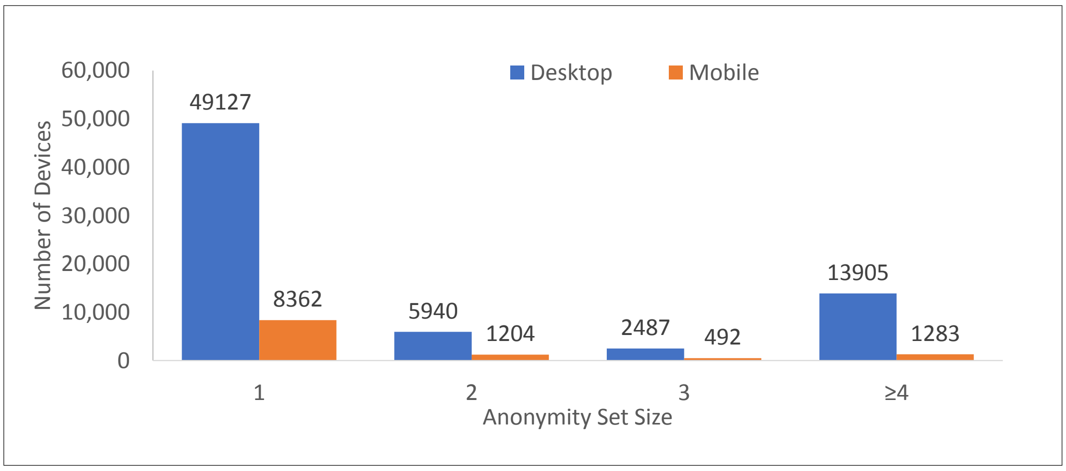


Figure 12: Number of devices in anonymity sets of size X (Desktop/Mobile)

fall within their own anonymity set for both platforms.

3.2.2 RQ2: iOS vs. Android: Which Platform is More Fingerprintable?

AIU Dataset: The comparison between iOS and Android as shown in Figure 13 is remarkable. The shapes of the graphs are striking by their similarity to the mobile and desktop results. These results are based on the AmIUnique.org dataset, and includes a wide variety of models of iOS and Android devices. Notably, one might expect iOS to have a greater number of large anonymity sets, due to the smaller number of models of devices that are extant within it (being limited only to devices produced by Apple), but proportionally, this doesn't prove to be true.

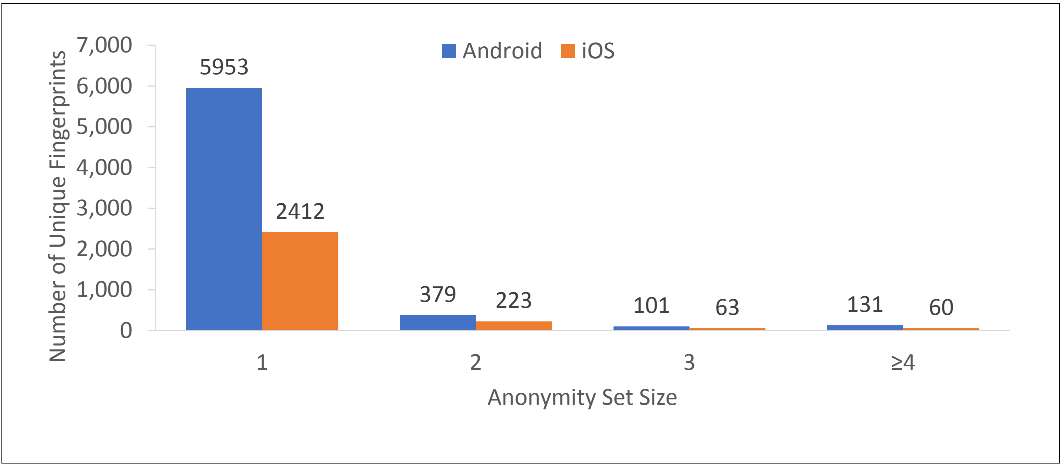


Figure 13: iOS Anonymity Set Sizes vs Android

In this case, more than 90% of Android and just over 87% of iOS anonymity sets, representing 75% and 71% of individual devices, respectively, are of size 1.

Taking a look at devices, we can see in Figure 14 that the graphs continue to reflect the same pattern between mobile and desktop, though once again this skews the results towards larger anonymity sets. Just over 2% of fingerprints were repeated 4 or more times in both Android and iOS, but these accounted for 12% and 10% of devices respectively, so although the numbers are very similar, it could be said that this suggests that iOS provides a marginally more private experience, but it should also be noted that the largest iOS anonymity set is only of size 17, whereas Android's largest is of size 32.

Other browsers on Android

Manually Curated Dataset: Within other—that is, non-Google Chrome—browsers on Android, our enquiries showed that the vast majority were simply Android WebView (identifiable by its user-agent). Of the 178 browsers that we looked at, 150 of them were merely Android WebView. 6 of those 150 appended something unusual or identifying to their user-agent string. All six did so to provide one or both of an identifier of the browser program, and a version string.

One of them, Amigo Browser,

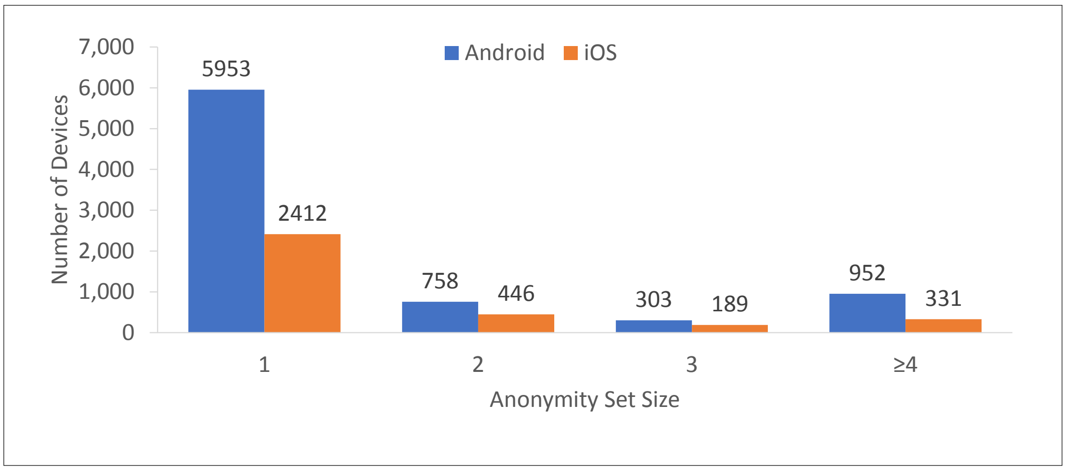


Figure 14: Number of devices in anonymity sets of size X (iOS/Android)

provided by mail.ru, went further; actually appending a UUID string to the user-agent, which is very peculiar and worth noting — it essentially fingerprints instances of the browser for you, meaning very little work is actually required by a website to fingerprint it. It is probably only used for mail.ru services, but it does seem a bit odd when a browser cookie would suffice, and would not also leak that UUID information to other web services. It would be very interesting to know if that UUID can be linked back to the computer in the Department of Computing where we ran our test, (but we haven't made any enquiries at mail.ru, and that is mere conjecture).

Putting that aside, we also took a more detailed look at a subset of browsers with the fingerprintjs2 tool. The browsers we tested were:

- InBrowser
- Firefox Focus
- Dolphin Browser
- Opera
- Brave
- Ghostery
- Firefox
- Microsoft InTune
- Yandex
- Chrome
- WebView Browser Tester

These were done on four different emulated devices — an LG Nexus 5

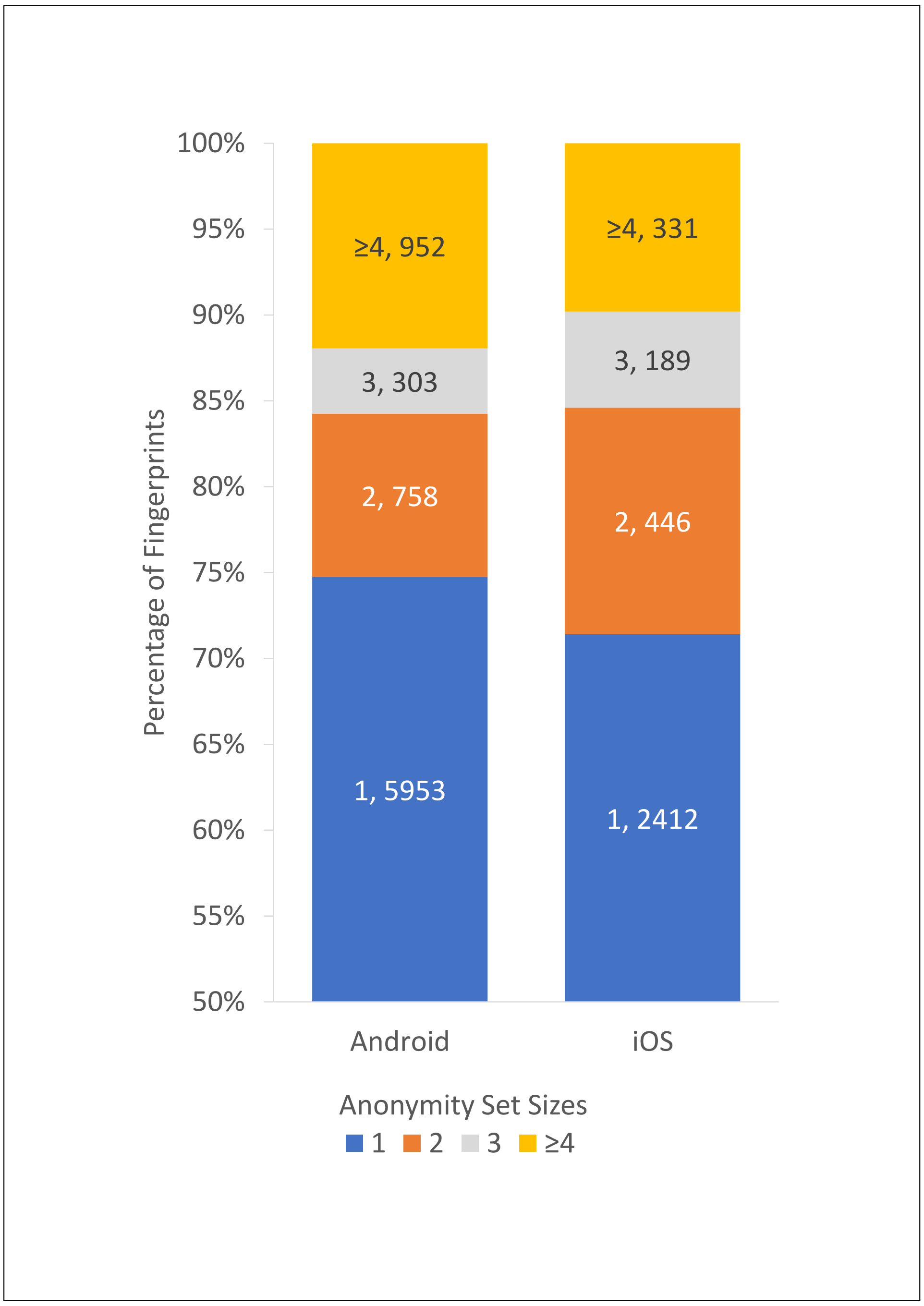


Figure 15: Percentages of fingerprints in anonymity set sizes for iOS and Android

```
Mozilla/5.0 (Linux; Android 8.0.0; Android SDK built for x86
Build/OSR1.170901.043; wv) AppleWebKit/537.36 (KHTML, like
Gecko) Version/4.0 Chrome/64.0.3282.137 Mobile Safari/537.36
```

Figure 16: Android WebView user-agent String, recognisable by the "wv" in the OS section

```
Mozilla/5.0 (Linux; Android 8.0.0; Android SDK built for x86
Build/OSR1.170901.043; wv) AppleWebKit/537.36 (KHTML, like
Gecko)
Version/4.0 Chrome/64.0.3282.137 Mobile Safari/537.36
AMIGOAPP UUID_0e0cc18ceabf47bef704fec82b8eb1ba APP_VERSION_-
1.10.187
```

Figure 17: Amigo Browser user-agent string

and 5X, with both Android 7.1.1 and 8.0.0.

What was striking was that between these (emulated) devices, there were no fingerprints that were repeated, but within each device/OS version, two browsers were identical to another browser each. These were Brave, which had the same fingerprint as Chrome (unsurprising as the former's code base for Android is based on that of Chrome for Android), and Ghostery Browser, which was identical to the Android WebView test program. We can only assume that this is because it is essentially the same but with tracker-blocking capabilities.

Another curiosity was the differences between Firefox and Firefox Focus, with the latter marketed as a privacy-conscious browser by Mozilla. They do not appear to share the same codebase (at least not in terms of JS or rendering engine), as Firefox Focus appears to use WebKit, whereas Firefox proper uses its desktop namesake's Gecko engine.

We believe that with the vast majority of Android users being Chrome users, with a small number of Firefox users, that anyone using a different browser will be fairly obvious. This disregarded other browsers, such as the webviews found within common applications, such as the Facebook Messenger webview, or the BBC News webview. However, these are not typically used beyond the page they initially visit

— though of course it is fun to attempt to break out of a webview by trying to reach google.com!

iPhone vs. iPad

AIU Dataset: Looking at Figure 18, we see the same data as Figure 14 for iOS, split into iPad and iPhone — which are the two devices that run iOS (as distinct from watchOS or macOS).

All that can really be said is that the iPhones appear to have something of an upper hand with regards to having larger anonymity sets (with their largest being 17, compared to the iPads" 11), but this should be tempered with the fact that there simply aren't very many iPads in the data.

The pattern is once again very similar in general to the previous patterns that we've seen, again repeating the shape of graph that we've already seen.

3.2.3 RQ3: What Sources of Fingerprinting Exist on Identical Device Models?

Manually Curated Dataset: We studied a small number of iPads provided by the Department of Computing. All four were the same model (2017 iPad Pro). However, only two of them returned the same fingerprint within Safari, despite them all running the same version of iOS (11.3). They were all also reset, wiping all settings and content before the fingerprint was collected.

If we look at Table 3, then we can see what the primary differentiators are:

- Do Not Track setting
- Language Setting
- Time Zone
- user-agent (and therefore Model Number / Hardware Revision)

AIU Dataset: We will pick a uniquely identifiable device model — in our case, the iPhone 7. We are able to do this as we know from Apple Data Sheets [34] that the iPhone 7 has the Apple A10 Chip (picked up as the "Apple A10 GPU" as the WebGL Renderer), and has a smaller resolution than the iPhone 7 Plus or the 2018 model of the iPad. However, we know that although the Data Sheet gives a resolution of

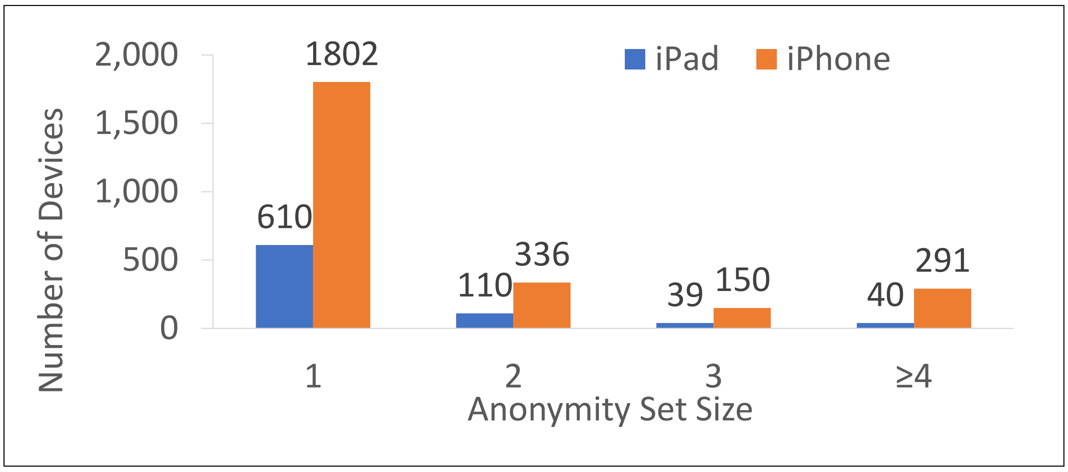


Figure 18: Devices per anonymity set of size X (iPad/iPhone)

Id	DNT	Language	tzOffset	userAgent
A	1	en-GB	-60	.../11.0 Mobile/15E148 Safari/604.1
B	0	en-GB	420	.../11.0 Mobile/15E148 Safari/604.1
C	0	en-US	-60	.../11.0 Mobile/15E148 Safari/604.1
D	0	en-GB	-60	.../11.0 Mobile/15D100 Safari/604.1

Table 3: Changed variables across DoC iPads

1334x750, the reported resolution by the browser is actually 375x667. You will note that this is exactly one-quarter of the above resolution, rotated by 90°. This is so that web pages display at an acceptable size on high-DPI (dots-per-inch) screens that typify smartphones.

We can then see in Figure 19, showing that 26 of our 57 iPhone 7s are unique, and that the largest anonymity set is merely of size 7.

Key Takeaway: It *is* possible to distinguish between individual devices in a group of identical ones.

We should also take a look at what these distinguishing variables are. In Table 4, we can see that just under half of the iPhone 7s are set to US English, and a further 20% are set to Brazilian Portuguese.

Key Takeaway: Language/locale setting is a very good discriminator on a per-device model basis.

We also have an interesting set of time zones (which show a large

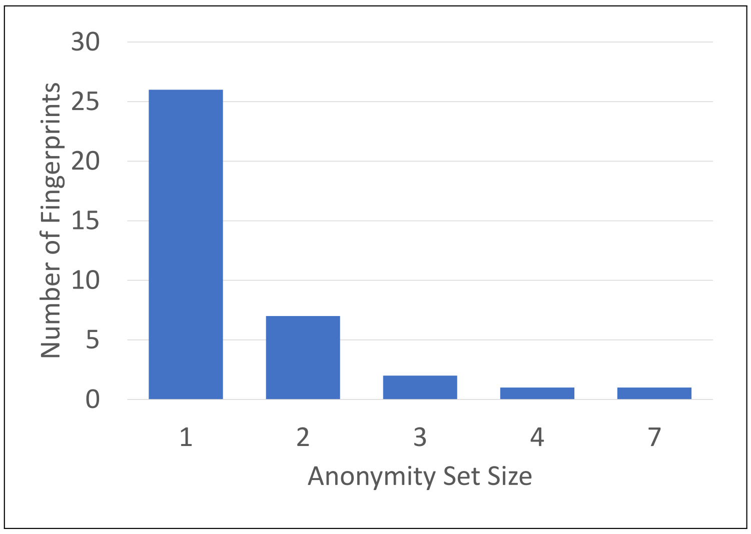


Figure 19: iPhone 7 Anonymity Sets

Language	Frequency
en-us	25
pt-br	11
fr-fr	4
en-au	2
zh-cn	2
vi-vn	2
en-gb	2

There are also 1 each of:
it-it, en-ca, tr-tr, nl-nl, ru, de-at, pl-pl, en-ie, pt-pt

Table 4: Language discriminators in iPhone 7

Time zone	Frequency
120	5
360	4
300	4
-60	4
480	3
-120	3
-540	1
420	1

Table 5: Time zones in the en-US locale

number of Brazilian users in DST (UTC–02:00). We also note that there are 3 devices shown in the '0" time zone, representing GMT/UTC, and this corresponds to the 3 devices using British (2) and Irish (1) English as their language setting — no other locales are mentioned in the language list that are in the GMT/UTC time zone. As the data was collected in the Northern Hemisphere's winter, this is expected. Note also the large number of Time zones within en-US; this acts as a further discriminator for that language, both splitting up actual U.S. users into the four time zones that make up the contiguous U.S., but also those users using en-US (as a default) elsewhere in the world. The breakdown of the en-US locale by time zone is given in Table 5.

Key Takeaway: Time zones are another excellent discriminator within a language, especially en-US, and on their own.

AIU Dataset: Roughly 60% of devices had no "Do Not Track" setting enabled, with the rest mostly requesting that they not be tracked, and two explicitly allowing tracking. It should be noted that according to Quantable [35], only around 15% of real-world web users actually use DNT (many of them Internet Explorer 10/11 users, which default to DNT: On). The AmIUnique data, on the other hand, shows a much higher percentage with the setting enabled, but this can likely be put down to the fact that such fingerprinting study tools are far more likely to be visited by the sort of people that already know that Fingerprinting and tools like Do Not Track exist.

Key Takeaway: DNT might be useful in helping to re-identify users who use it, but is probably not useful in identifying a user/browser outright.

3.2.4 RQ4: What Different Browser Fingerprinting Vectors Exist on Mobile?

Mobile Browsers generally have the same capabilities as their desktop equivalents: in Mac OSX Safari for iOS; and on Android: either Google Chrome or themselves, in the case of browsers like Firefox which implement the same rendering engine. This means that they can easily collect a large number of data points as we've discussed above, including:

- IP Address
- Operating System
- Browser Name and Version
- System Language
- System Time zone (relative to UTC)
- HTTP Referer (previous page if clicked by hyperlink)
- Installed plugins and versions
- Screen Resolution
- Local Data Storage configuration
- Installed Fonts
- Execution profiles via HTML5 Canvas and WebGL APIs
- Additional data from specific plugins/extensions

Some other work then goes into further details, especially about the last two points: execution profiles, and data from specific plugins or extensions. In particular, Mowery et al. [36] was able to use the SunSpider benchmark and V8 benchmark [37, 38] to gain further information about the performance profiles of particular browser, allowing them to identify a browser version even if the user-agent was being faked by the browser.

Key Takeaway: It is still possible to correctly identify browsers to the OS and Browser Name/Version level, even if the user-agent is false.

If we want to be more specific to mobile, we can look at the various HTML5 APIs. However, many of these have been carefully designed in order to avoid giving away too much information, and mobile platforms won't pass data through unless that browser is actually the active application. In some cases, browsers will need to seek specific permission — from the user via the operating system — in order to use certain APIs, such as the Location API. However, others such as the accelerometer, do not require such permission. Some studies have been done in this area are showing promise that this may become very real in the future, not only as a fingerprinting technique with concerns for user privacy, but also as a method of authentication or as a second factor for authentication [5, 39, 40].

In order to build up a picture of a user using, for example, the gyroscope, the collection utility would need to run constantly in the background (for example, Android has a feature that on some phones allows it to be unlocked by fingerprinting a user's walk's effect on the gyroscope).

However, using momentary requests, like Location, might be more useful for fingerprinting someone if they tend to use a device in the same place repeatedly (though such devices tend to keep the same IP address). Again, this can't track a user's location over time without the user actually visiting the web page(s) that collect the data.

Key Takeaway: Some HTML5 sensor APIs can be used for relatively effective fingerprinting, but are not as straightforward to use as more traditional fingerprinting methods. Other APIs are not very useful, or at least, not useful if fingerprinting is to be done transparently (or secretly).

If we actually ask ourselves what these different fingerprinting vectors on mobile devices are, it is clear that the answer is the access that is available to physical sensors that aren't typically available on desktop devices. The utility of these in recognising users is emerging, and as we've discussed in previous sections, standard desktop-based fingerprinting methods are still sufficient to uniquely identify a high proportion of devices. Further, these methods are more useful for identifying a *user*, rather than a device, and are more useful in the context of a native application than a web browser. However, they shouldn't be discounted as more data is available to web applications.

3.2.5 RQ5: Is Fingerprinting Detection Easier to Evade on Mobile Devices?

Browsers in general seem to have a very limited set of options when it comes to defeating fingerprinting. The primary recourse, judging from our own data collection, is that they disable JavaScript, which is actually quite reasonable from a privacy point of view. However, many websites will not function correctly, and in the Web 2.0 world, where JavaScript is

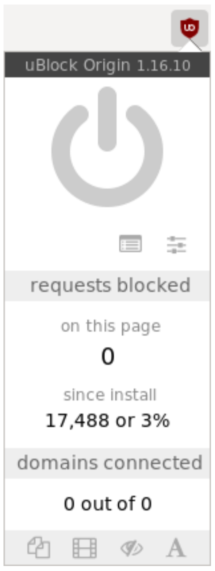


Figure 20: uBlock Origin's controls are not very granular

prevalent on websites to provide a good user experience, this is simply not acceptable to most users. Some blockers, like uBlock Origin [24], when disabled by users for this purpose, turn themselves off entirely for that website/domain, and therefore provide no fingerprinting protection at all. Others, like NoScript [41] (Figure 27), provide a granular set of permissions that can be applied at the domain level (though not at the individual script level, though this can be acheived with Adblocking lists used in extensions like uBlock or AdBlockPlus [10]

Manually Curated Dataset: In general, browsers returned one of two things; either the correct information, or no information at all, as the information is nearly all collated by JavaScript. In the modern world, if we want to make much sense out of it, we'd probably need to as well, though we could collect a subset of the data (such as user-agent, other browser headers, and the UUID cookie) using PHP (or another server-side language) only. Of course, whilst this could be considered evasion of fingerprinting, it's not hugely effective — very few web users actually disable JavaScript nowadays, and so it stands out like a sore thumb in the results. Of course, many online trackers don't submit data using POST requests like we did, so they might not show up in real-life datasets.

We here discuss "normal" browsers by operating system, and then some browsers specifically designed for use with the Tor network.

iOS

In our iOS investigation, we studied both Firefox and Safari running on a

DoC iPad, with and without private browsing mode, and in the case of Safari, with various content blockers, as well as the Brave web browser. Firefox was notable as the iOS version does not support content blockers, and even with its tracking protection feature turned on in private browsing, did not produce a different fingerprint.

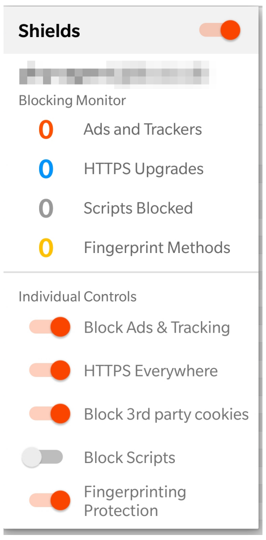
Safari on the other hand, did manage to change its fingerprint in private browsing mode, but this was very conspicuous — the only change is that the "Do Not Track" header was explicitly enabled. This is a peculiar default behaviour as it immediately singles out anyone using private browsing mode on Safari (which does not seem to have DNT turned on by default).

With content blockers in their default state, the fingerprint data sent from Safari was identical, though it should be noted that when AdGuard (one such blocker) had its tracking protection turned up, it did block our fingerprint script from running and so returned empty results. Something interesting to note is that all of the iOS browsers that did return results for fingerprinting all had identical canvas values. This reflects the fact that Apple requires all web browsers on the platform to use the platform-provided rendering engine (i.e., that of Safari, that being Apple WebKit).

Key Takeaway: Blocking of the JavaScript that performs fingerprinting is probably sufficient, as a lot of data is transferred via AJAX rather than HTTP POST requests. This should not translate into complacency however, as there are ways other than AJAX to get that data back.

Brave for iOS successfully defeats the fingerprinting script by blocking its execution, but not by default; only when the appropriate "shield" is enabled. However Brave for iOS, which is based on Firefox, has a peculiar addition to the user-agent string, which seems to identify the particular tab that a particular page is being opened on. Other than that, the instances of brave were not distinguishable from each other. However, they were distinguishable from Safari on iOS, which is potentially problematic if our goal is to be in the largest anonymity set.

Key Takeaway: Brave successfully



Shields ●

[redacted]

Blocking Monitor

0 Ads and Trackers

0 HTTPS Upgrades

0 Scripts Blocked

0 Fingerprint Methods

Individual Controls

● Block Ads & Tracking

● HTTPS Everywhere

● Block 3rd party cookies

○ Block Scripts

● Fingerprinting Protection

Figure 21: Brave features an array of different privacy options

defeats fingerprinting when it promises to do so, but again does this by preventing the script from execution, rather than by providing false information.

Android

On Android, we've studied four browsers, two of these being stalwarts Google Chrome and Firefox, and the other two being privacy-focused Firefox Focus, and Brave.

Something notable is the way the collected canvas data clearly shows the rendering engines that each browser is based on: Firefox using its own Gecko engine; and Chrome, Brave, and peculiarly, Firefox Focus, using Blink (based on WebKit). Indeed, Brave does a good job of impersonating Chrome, and at the time of our test only gets a different fingerprint hash because it did not quite have the same version number as Chrome in the user-agent — though an earlier test showed both with the same version number. This is a symptom of software development cycles with rapid-release though, rather than a mistake on the part of the developers, and is not likely to make a browser more fingerprintable. When we used Brave's "fingerprinting protection" mode, it no longer allowed our fingerprinting script to run, thus preventing fingerprinting. It's also been the only browser that makes direct reference to fingerprinting by name.

Firefox Focus, on the other hand, does not succeed in blocking fingerprinting at all, and in fact its fingerprint is identical in both its "ordinary" mode, and it's "disable tracking" mode, which is a little disappointing, and does not appear to attempt to make the user blend in

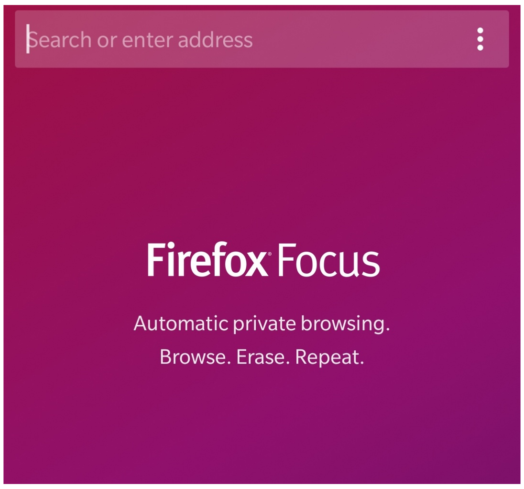


Figure 22: Firefox Focus automatically erases browsing history

in any particular way. Even more unhelpfully, as it's not a very common browser, its user-agent clearly identifies that Firefox Focus is being used, which will show it up very clearly to anyone trying to fingerprint a user across the web.

Firefox proper for Android, unlike its iOS counterpart, is able to make full use of its own rendering engine, and has support for add-ons. We've tested it in and out of private browsing mode, and with popular extensions uBlock origin, and AdBlockPlus. In almost all of our tests, Firefox returns the same results each time, with the singular exception of uBlock Origin being enabled, which successfully blocks the use of the fingerprinting script. It is unfortunate that Firefox's private browsing mode, which touts "Tracking Protection" amongst its various features, does not successfully block the fingerprinting script for what it is.

Google Chrome for Android does not presently support extensions or add-ons, and so the only available modes are for ordinary browsing, and private browsing, known as incognito mode. Both modes produce a full fingerprint, and it is identical between them. However, unlike Firefox, Chrome does not advertise "Tracking Protection" in its private browsing mode, and so this is not an unexpected result.

Key Takeaway: Fingerprinting protection for major browsers on Android is lacking, though Brave will successfully recognise and block fingerprinting — or at least, the fingerprinting we were doing — from taking place.

Desktop

On the desktop, we tested both Chrome and Firefox running on

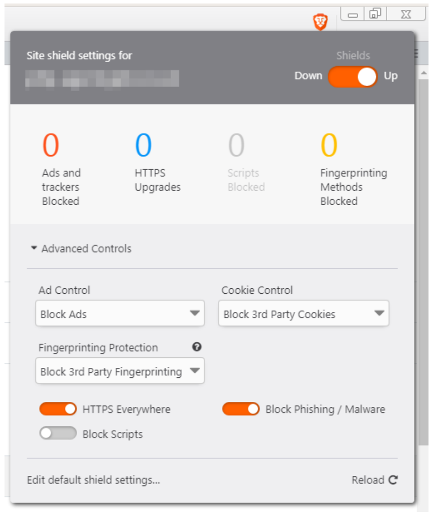
Windows 7, with some different extensions.

Firefox on Windows 7 windows 7 behaved similarly to Firefox for Android, though generated a different canvas image value. We tested it with uBlock Origin and with NoScript. We achieved a full fingerprint from Firefox in normal and private browsing mode, and with NoScript enabled but not blocking JavaScript from the domain of the fingerprint tool — by default it allowed JavaScript from the CDNs that the fingerprinting libraries were running from. With uBlock Origin, however, the fingerprinting script was blocked, and thus no information was collected.

Chrome, with combinations of AdBlockPlus and uBlock Origin, always returned the same fingerprint. This is peculiar and worth noting, because uBlock Origin has otherwise blocked the tool from running, but it did not do so in Chrome. This seems like a peculiar behaviour, as one would expect it to exhibit the same behaviour across browsers. This may be due to a slightly different default filter list setting, or perhaps down to a quirk in the way the Chrome desktop version of the extension is written. The only extension that successfully blocked fingerprinting was noJS, but as that is merely a simple switch that prevents any JavaScript from running, this would not appear to be fingerprinting-specific.

Brave, this time for Windows rather than for Android, performed well. Once again it is very similar to Chrome, but there are some differences (which would be out of place for a "true" Chrome instance). The major one is that the Chrome PDF plugin does not come with Brave, or rather Brave does not identify that it does. By default, this plugin is enabled in Chrome and would be reported by the fingerprinting tool. Whilst it's arguably more private not to reveal any plugins, we would argue that it makes more sense for Brave to imitate Chrome as much as possible, including by possibly falsifying the list of installed plugins to include the Chrome PDF viewer.

On the desktop, Brave has the option to allow "all", "none", or "all except third-party" fingerprinting. In our test, the fingerprinting library we are



Site shield settings for [redacted]

Shields Down Up

0 Ads and trackers Blocked

0 HTTPS Upgrades

0 Scripts Blocked

0 Fingerprinting Methods Blocked

Advanced Controls

Ad Control: Block Ads

Cookie Control: Block 3rd Party Cookies

Fingerprinting Protection: Block 3rd Party Fingerprinting

HTTPS Everywhere: on

Block Phishing / Malware: on

Block Scripts: off

Edit default shield settings...

Reload

Figure 23: Brave for Desktop has a slightly different set of options compared to Android

using is technically from a third-party domain, being on the CloudFlare CDN. However, we suspect that well-known CDNs are on some form of third-party whitelist, as scripts on those sites are used in a vast number of websites. Curiously, changing the shield setting between allowing all and allowing all except third party fingerprinting sometimes caused the WebGL value to change. This is a very peculiar result and Brave may wish to investigate how this occurred — though of course it could also be down to other factors, such as the VM it was running in, the graphics drivers, or, everybody's favourite, cosmic rays.

Key Takeaway: Fingerprinting on mainstream desktop browsers is almost always possible, though some extensions on certain browsers or in certain configurations will block fingerprinting JavaScript from execution. Brave also blocks some fingerprinting, but is also not perfect and when it does not block fingerprinting, it gives way to suggestions that it might not be an instance of Chrome.

In the next section we'll look at the Tor browser and how it compares to its desktop rivals.

Tor Browsers

The Tor project itself provides an official browser only to Desktop devices, though Android is also provided with the Orbot Tor client and Orfox browser by the Guardian Project, which is endorsed by the official tor project. On iOS the field is less clear, with browsers being provided by unknown (and therefore not necessarily trustworthy vendors, including some that charge a subscription fee to connect to `.onion` hidden services, or even to be used at all. The closest thing to Orfox on iOS would appear to be the "Onion Browser", which claims to have support from the Guardian project (who create Orbot and Orfox for Android).

Tor on the Desktop

On desktop devices, we ran the Tor browser on two computers, one running Windows 7 and the other running Ubuntu. We also set up our fingerprinting tool as a hidden service so that it could be accessed (behind a firewall as it is) from the Tor network. First, without maximising the window, we tested the browser on its three default security levels, which do the following:

Standard All Browser and website features are enabled (except plugins, which aren't available in the Tor browser)

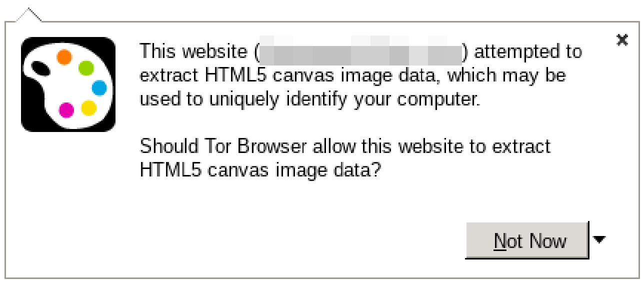
Safer Disables some features that could be dangerous

- Disables JavaScript on non-HTTPS websites (by default).
- Some web fonts and mathematical symbols are disabled.
- HTML5 media (audio and video) are click-to-play.

Safest Only allows features required for static sites by default.

- Disables JavaScript on all sites (by default).
- Some fonts, icons, mathematical symbols and images are disabled.
- HTML5 media are click-to-play.

In the first security level, we were able to fingerprint as normal, though


This website ([redacted]) attempted to extract HTML5 canvas image data, which may be used to uniquely identify your computer.

Should Tor Browser allow this website to extract HTML5 canvas image data?

Figure 24: Tor Browser blocks canvas data collection by default

we were prompted about the collection of canvas data (Figure 24). In the second and third level, we had to explicitly allow scripts for any form of fingerprinting to take place. In all three security levels, we had the same fingerprint result; until we maximised the browser, when a different result was given (due to the resolution change). All of the Tor Browsers that blocked the collection of canvas data returned the same value for the canvas data, which means that it successfully works across platforms.

However, there were some discrepancies. Although the Tor Browser does its best, bearing in mind that it is cross-platform (based on Firefox ESR), with identical results for platform (Win32), user-agent (Firefox 52 on Windows 7), and for other things like colour depth, the fonts list differed between the Windows and Linux versions, with the Windows version producing a list of fonts clearly identifying it as Windows with fonts such as MS Gothic and Segoe UI, whereas the Linux version returned no fonts.

Key Takeaway: Tor Browser on desktop does a good job of obfuscating and hiding fingerprintable characteristics. However, it's not perfect and some platforms will still return slightly different results.

Tor on Android

Running Orfox on Android, which connects through Orbot, the default position is to block the execution of the fingerprinting script, and as such no results are returned. As we've discussed, most tracking is done through JavaScript-only, and therefore this likely isn't a concern.

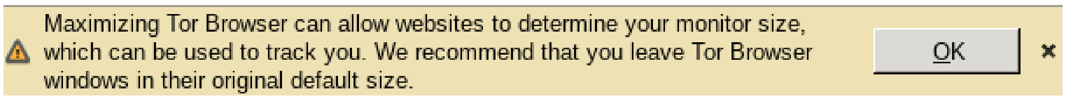

Maximizing Tor Browser can allow websites to determine your monitor size, which can be used to track you. We recommend that you leave Tor Browser windows in their original default size.

Figure 25: Tor Browser warns users who maximise it against tracking

However, if the user manually intervenes and enables JavaScript, then the full fingerprint script can run. Pleasingly, it returns the same HTML5 canvas data as the Tor browser on Windows and Linux (when blocking canvas data). It does however return a resolution consistent with the viewport — this matches the behaviour of the Tor browser in general, but a limitation of mobile devices is that all phones of the same model and DPI setting will return the same result for the resolution. and identifies itself as an Android device running an ARMv8 processor. This differs from the Tor Browser which always self-identifies as Windows 7 and does not disclose the type of processor (though it can probably be safely assumed it's Intel x86_64 — though some new Windows 10 computers actually do have an ARM processor).

Key Takeaway: Tor on Android in the form of Orbot and Orfox provides a reasonable level of protection, but would only be fully anonymous amongst other Android Orfox users on a device with the same screen resolution and DPI setting.

Tor on iOS

iOS was somewhat disappointing in some ways, though potentially good in others. We tried a number of different Tor browsing apps, specifically: Tob, Torr, Evil Onion, Onion Browser, and "Tor Browser"[1].

Torr and Evil Onion both require various levels of subscription (the former to connect at all, and the latter to connect to `.onion` addresses), and so like earlier sections, we will discount them as we only want to look at free-of-charge browsers. Of the remaining three, "Tor Browser" does not successfully connect to the Tor network — we can discount network problems here

[1]Not to be confused with the desktop Tor browser: the app store and iOS interface names are different — in the App Store it is referred to as "TOR DeepWeb Browser by Art Fusion"

as both Tob and Onion Browser succeed.

Tob does not appear to run the fingerprinting tool at all, no matter what the settings are regarding the blocking and unblocking of ads, trackers, whitelists, etc., so it always returns a blank result. As we discussed earlier, this is probably not a bad thing. Tob also crashed quite regularly, though, so whether or not this is by design is questionable. Onion Browser appears to be the pinnacle of Tor browsers available on iOS, providing a fairly decent user experience and various customisable options (including the concept of security levels that we saw on the desktop). Its default and permissive modes both permitted the fingerprinting script to run, and it returned the same results for both. It was identified, like all the other iOS browsers we've looked at, as Mobile Safari, and had the same canvas data corresponding to the particular iPad hardware we were using. In fact, the only difference in the collected data between Onion Browser and Safari itself is that the former appears to append some numbers to the user-agent — though it is unclear what these numbers refer to — they don't match the version or build numbers found in the "About" page of the app, or in the App Store. When run in strict mode, Onion Browser blocks all JavaScript from execution, and therefore no data is collected from the fingerprinter.

Key Takeaway: Although Onion Browser seems to be fairly good at anonymisation, it still has a slightly distinguishing user-agent, which means it could be identified in a crowd of ordinary Safari users. The general offering of Tor on iOS still seems to be poor and in need of further development.

Subverting extensions

Mowery et al. [36] were also able to subvert extensions in the use of fingerprinting, in this case the NoScipt extension [41] (Figure 27), which at the time of writing has almost one million users. Of interest is that Mowery et al. were able to subvert the whitelisting mechanism of NoScript, by making requests to the Alexa Top 1000 domains. They were able to identify 689 sites with suitable JavaScript that when fetched

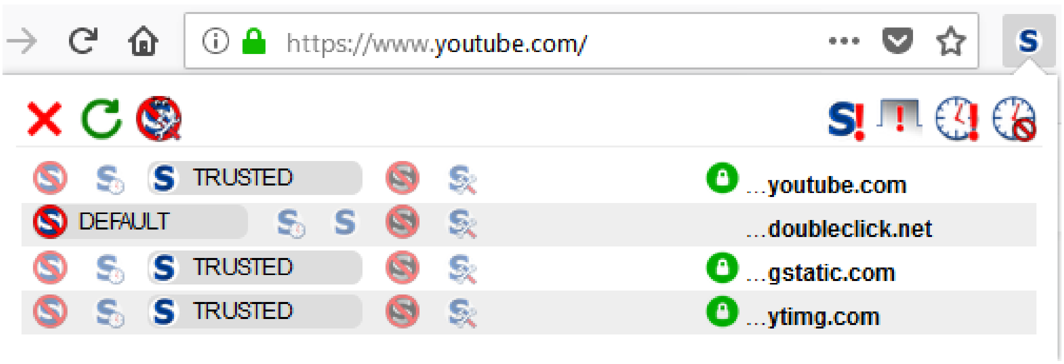
```
Mozilla/5.0 (iPad; CPU OS 11_2_6 like Mac OS X)
AppleWebKit/604.5.6 (KHTML, like Gecko)
Version/11.0 Mobile/15D100 Safari/604.1
```

(a) Mobile Safari

```
Mozilla/5.0 (iPad; CPU OS 11_2_6 like Mac OS X)
AppleWebKit/604.5.6 (KHTML, like Gecko)
Version/11.0 Mobile/15D100 Safari/604.5.6/7651480384
```

(b) Onion Browser

Figure 26: Comparison of Mobile Safari and Onion Browser user-agents



https://www.youtube.com/

TRUSTED			...youtube.com
DEFAULT			...doubleclick.net
TRUSTED			...gstatic.com
TRUSTED			...ytimg.com

Figure 27: The NoScript Browser Extension running in Firefox

wouldn't cause problems (like web browser crashes, or `alert()` calls). It took around 22 seconds to test all scripts, but it could do 95% of them within 1500ms (which would be the case if NoScript were disabled). It should also be noted that NoScript actually prevents HTTP requests to `GET` the blocked scripts. With NoScript turned on and blocking, the vast majority of domains could be checked within 100ms.

Key Takeaway: It is possible to use leverage popular extensions to increase fingerprinting efficacy.

Because NoScript is favoured by privacy advocates, it is possible that in places where surveillance of "subversives" is common, that this method could be used to expose them, as NoScript whitelists tend to be fairly unique across all sites (the author speaking as a user of NoScript). It is possible that such a technique is used by law enforcement agencies both here and abroad

— though this is mere speculation.

Key Takeaway: Ironically, privacy extensions like NoScript can actually increase the effectiveness of fingerprinting.

Answering the Question

We started this section by asking: on which platform is it easier to evade fingerprinting. In my view, this is still the desktop's domain — though we look forward to up-and-coming browsers like Brave turning this around. Far and away the best (and most anonymising experience) seems to be the Tor browser, running on Windows, in terms of being in as large an anonymity set as possible. However, we would also say that Brave, should it make some tweaks in the way it reports plugins, may well leap past the Tor Browser in the same terms. However, some of these, such as enforcing window sizes, are not especially user-friendly, and so may not be especially popular. Mobile in general seems to be pretty poor at preventing or defeating fingerprinting, though as we mention

Chapter 4

Related Work

By far, the most closely related study to ours in the literature is Laperdrix et al. [26]. It is based on the dataset of AmIUnique.org at the time of publication (and the full dataset rather than the subset we are working with). They also discuss the future of fingerprinting, and whether or not it will become more pervasive and effective in the future, or less so, with the rise of new technologies like HTML5, and the decline of plugins such as Adobe Flash.

4.1 Studies of Fingerprinting

Fingerprintability of devices: The team behind AmIUnique.org [3] gathered 118,934 fingerprints composed of 17 attributes using recent web technologies for their main study based on the entire dataset. HTML5 provide access to many parts of a system, notably with the use of the Canvas API which relies on multiple components, hardware and software, working together. They also show that browser fingerprinting is as effective on mobile devices as it is on traditional desktop-form devices. They also evaluate how browser fingerprinting could stop being a threat to user privacy if some technological evolutions continue (such as the long decline of browser plugins like Adobe Flash), or the incorporation of new technologies into web browsers by their various creators. [26]

Execution characteristics of JavaScript engines and fingerprinting of extensions: In 2011, Mowery et al. looked at two possible methods of fingerprinting; by studying of a browser's JavaScript engine's execution characteristics, which are hard to imitate or fake; and attempting to work out what a user's NoScript whitelist is — though this relies on the user having that particular extension installed. Seven years on, in 2018, we could actually do something similar using AdBlock lists and the Ghostery configuration to fingerprint request-blocking extensions as well as the browser they are running on. [36]

HTML5 APIs: The ability of HTML5 to track users is brought up by Princeton's Arvind Narayanan, and referenced in *The Register*, whereby users could be fingerprinted using novel APIs now available to web developers that were previously only available in the realm of native applications. These include audio playback, where different browsers played audio with slightly different characteristics, and when combined with other new indicators, such as the gyroscope of a mobile phone, we may see in the future detailed fingerprints being made possible using only HTML5. [4, 42]

iOS app fingerprinting: The authors of Kurtz et al. noted in 2016 that Apple removed access to various device hardware identifiers that were frequently misused by iOS apps to track users. They studied just how much they were able to glean from the operating system despite this. Using Apples iOS as an example, they were able to gather 29 different individual data points per device. These features can be queried from arbitrary third-party apps via the official SDK. Their data, comprising 13,000 fingerprints from approximately 8,000 different real-world devices show that all these native app-based fingerprints are entirely unique, and were able to use a supervised machine learning technique to correctly re-identify a device even when its fingerprint had changed in 97% of cases. Notably, this work demonstrated just how much information is available to native iOS apps, including the user's music library and no. of plays for each song. Figure 4, which is taken from this paper, shows the fingerprintable features, and how much they change. [6]

Fingerprinting of devices by malware: Bulazel et al. make the point that much malware uses fingerprinting techniques to determine if it is being analysed or not — for example, can it detect a virtual machine hypervisor, or is this somehow recognisably part of a malware analysis tool like Cuckoo, or does it exhibit strange behaviour that would only happen in a laboratory? They discuss techniques used by malware to evade automated analysis through the use of fingerprinting (including environmental artefacts — such as system settings or files — as well as Timing Attacks, such as checking how long certain operations take to run, which can be detected in JavaScript in a web browser, and also other peculiarities associated with Virtual Machines, such as strange CPU instruction execution results or emulated device sensors in the case of mobile devices. They also discuss how to detect and evade this evasion, thus allowing the malware to be properly analysed. [43]

4.2 Preventing Fingerprinting

Fingerprintability of browsers: In Vastel et al., they presented FP-TESTER, which is an approach to automatically test the fingerprintability of web browsers, and they implement a toolkit for doing so. They also look into the effectiveness of countermeasures that may be taken by a browser natively or by an extension or add-on, and into whether the side-effects of that attempt to block fingerprinting or tracking actually make the browser more identifiable. They also look at how such countermeasures impact the amount of time it takes for a fingerprinter to be able to gather enough data for a fingerprint. [44]

Private browsing privacy: Wu et al. compared the 'Private Browsing' modes available on popular desktop and mobile browsers, finding in their results many disagreements between browser vendors as to what would be blocked or changed, and inconsistency in the same vendor's approach to desktop and mobile browsers. Some of these are a trade-off between privacy and security, but they did find that even if private browsing mode leaks no information about the *user*, an attacker (fingerprinter) could still fingerprint the browser and trace the user that way. It had been suggested that browsers could defeat fingerprinting by reporting random data for such attributes as installed fonts and plugins, but in this paper, they authors showed that such an approach is insecure, via an attack based on statistical methods — which they found to be both easy and effective as an attack. [45]

Quantifying tracking protection tools: Merzdovnik et al. quantifies the effectiveness of tracking protection tools at scale. They analyse the architecture of various solutions and compare them. and study on effectiveness of popular tracker-blocking tools. They are able to quantify through their analysis the protection that tools give against trackers present on more than 100,000 popular websites and 10,000 popular Android applications. They also provide some insight into the ongoing "Ad Wars." Among other things, they discover that rule-based browser extensions give a better performance than ones based in machine learning (which is interesting for any company or person who uses primarily human-curated rules as a primary method of classifying things), trackers with smaller footprints (such as uBlock Origin [24]) are more successful at avoiding being blocked, and there is great concern about the role that CDNs are playing and will play in the future with regards to tracker-blocking tools. [46]

4.3 Tracking users Online

Desktop tracking: In Sculley et al. of Google, large scale data mining effort is presented that detects and blocks such adversarial advertisements for the benefit and safety of Google's users. Due to the cost of both false negatives (danger to users) and false positives (loss of revenue to Google), the system combines both automated and semi-automated methods in order to learn from the highly skewed data that their systems collect. To do this, they bring in humans, with domain expertise, and independently assess the system's effectiveness.[47]

Person-centric tracking: In Zimmeck et al., the concept of *person–centric*, as opposed to *application* or *device* –centric tracking is introduced.

Person–centric tracking attempts to instead attempts to track a person across all their devices and applications. They show that they are able to match desktop to mobile devices with an F_1 score of 0.91, and highlight the increasing accuracy this is likely to give way to as machine learning comes to be more prevalent amongst technology and advertising companies. [48]

Privacy in Mobile: Apps vs Browsers: Papadopoulos et al. ask whether or not using a mobile native app as opposed to a mobile browser is the best option for an individual's privacy. To do so they study a number of apps to search for privacy leaks, using a man-in-the-middle method to inspect traffic from the app. They then implement a mechanism called *antiTrackDroid* that works to try and prevent tracking, enabling users to access online services via a mobile app without risking their privacy. The system reduces the leaking identifiers of apps by nearly 30% whilst imposing a virtually negligible impact on latency. [49]

Use of user profiles in advertisement selection: Barford et al. study the correlation between certain advertisements and user profiles, with over 175,000 distinct adverts collected by their crawler. Despite the amount of tracking and profiling that advertising companies do online, they found that the variety of adverts shown to a user across all websites was much greater than those shown to all users of a particular website. However, they did find that in general the user profile was the most important factor in determining a particular advert, rather than the site itself. They also make observations about how online advertising has advanced from its infancy in the 1990s to today. [50]

4.4 Online advertising and ad-blocking

Stealthy blocking of adverts: Storey et al. implemented some new methods for adblocking, including methods to try and disguise the presence of adblocking (or lack of presence of adverts) from publishers. It does this by, among other things, borrowing ideas from rootkits and replacing some calls to the DOM with functions of their own in order to disguise the fact of adblocking from publishers. They also evaluate the legalities of adblocking (both blocker providers and users), and discuss where the arms race between adblockers and publishers is heading. They challenge the assumption that many (including me) have made, which is that the arms race will escalate indefinitely, but will instead settle down in the near future. [28]

Effectiveness of adblock filter lists: Iqbal et al. take a retrospective look, using the Internet Archive's Wayback Machine and historical versions of adblock filter lists, at the effectiveness of those lists against websites at the time of that version of the list, over the five years to 2017. They find that the effectiveness of those lists has increased significantly since 2014. They also detect anti-adblock scripts running on about 9% of the Alexa Top 5K websites. [13]

Effectiveness of Adblockers: In a world where many internet sites have relied on advertising for their revenue, they run the risk of running too many advertisements. This makes users far more likely to enable an ad-blocker, which then means that the publisher loses out on revenue. Garimella et al. study the performance of a number of popular adblockers on news websites, and the effect of adblockers on both user privacy and browser performance. [12]

Acceptable Ads Program: The *Acceptable Ads* program was created in order to encourage advertising that isn't annoying to users, and does this by whitelisting advertisers and websites that have appropriately non-intrusive advertising. Walls et al. have studied the growth of the program, and show that currently the acceptable ads whitelist of 2015 (when the study was carried out) is triggered (that is, overrides the blacklist to allow some adverts) on 59% of the Alexa Top 5,000 websites. They also suggest some ways to make the process more transparent, and criticise it for at times being opaque. [11]

Monetising sites in spite of Adblocking: Vratonjic et al. study the different ways that sites monetise themselves (e.g. ad-supported, freemium, premium sections, paywall) and study the best way to monetise a site to maximise return, in the knowledge that some users will use an adblocker. It would be preferred for all users not to use adblocking, as then the free, ad-supported model for content will work very well, but knowing that they don't, the study authors show that the "best" way (that is, maximise the utility function of revenue) to monetise a site is to do so differently depending on the user, and that one single monetisation technique for all users on a site may not be the best method of generating revenue. They also show that it's hugely important for companies to know both about users who may block ads (because they don't like them), and also about how much consumers value the content that is being monetised (i.e. is it *worth* paying for). [9]

Blocking the ad-blockers: The arms race between adblockers and publishers trying to evade blocking had only intensified. Nithyanande et al. study the way that third-parties are shared across multiple websites (enabling tracking), and they study the presence of attempts to evade ad-blocking across websites, finding that the scripts they were studying could be found on nearly 7% of the Alexa Top 5,000 websites, though potentially many more might have been using some form of anti-adblocking. They also find that in half of cases, at least one of the three adblocking browser extensions they studied — AdBlockPlus, Privacy Badger, and Ghostery — was able to itself avoid the anti-adblocking techniques being used by websites. [17]

Identifying anti-adblockers: In this study, by Mughees et al. in 2017, the authors use a machine learning-based approach to identifying anti-adblocker scripts running in the wild, and identify 686 websites in the Alexa Top 100,000 websites that change their page content if ad-blocking is detected. They look at the spectrum of responses, which range from a small message asking users to disable adblocking on their site, to denying any access to the site at all until adblocking is disabled. Unlike Storey et al. [28] they suggest that this "cat-and-mouse" game of escalation between the publishers and the adblockers will continue. [16]

Adblocking in the Wild: In a study of HTTP header information taken from a "Major European ISP", Pujol et al. analyse the usage of adblocking tools in the wild, and leverage AdBlockPlus to identify when adblocking is or is not taking place, combined with their own system for fetching web pages (to identify when ads ought to be being loaded). They find that a large minority (22%) of active web users in this particular ISP's residential broadband network use an adblocker when browsing, but in the context of AdBlockPlus, most do not disable the Acceptable Ads whitelist, which is a very useful thing to know. [8]

4.5 Mobile Device Fingerprinting

Fingerprinting using device sensors: Bojinov et al. did an extensive study into over 10,000 mobile devices, running both iOS and Android, using physical sensors and their imperfections to re-identify devices based on fingerprints from those physical sensors. They found that using a *combination* of these sensors in concert allowed them to re-identify devices with a high degree of accuracy, *even after a hard reset* of the device, in which all data is deleted. [39]

Fingerprinting using device sensors: Das et al. carried out a similar study to Bojinov et al., with the idea of using hardware sensors such as the accelerometers and gyroscopes in modern smartphones to generate fingerprints, alongside inaudible audio stimulation. They were able to generate a fingerprint in around 30-40 seconds, which could itself be used to identify a user within 5-8 seconds. [40]

Applying desktop fingerprinting techniques to mobile devices: Hupperich et al. study the applicability of "traditional", or desktop-based fingerprint techniques to mobile devices, and furthermore attempt to show that it is possible to build a fingerprinting system for extremely similar devices such as mobile devices, rather than the much more customisable and varied world of desktop machines. They also investigate how users may evade fingerprinting techniques. [51]

Authentication using device sensors: Wang et al. investigate the possibility of using the sensors on mobile devices for user authentication, in such a way that is simple for the user to use, but hard for an attacker to bypass. They use the geometry and movement of the user's hand, using the touch input and the gyroscope of the mobile device. In a study of 70 subjects, they were able to achieve a false negative/positive rate of only 2.5%. [5]

Chapter 5

Conclusions

In this project, we've used a mix of analysis of historical data and proactive collection and curation of specific device data. The historical data we've taken from the AmIUnique project, informed by the related work, and with a small implementation we've been able to collect some relevant data of our own, using a small number of devices that we've had access to. In general, we've found that a surprising number of identifying features could potentially be used to re-identify users if collected by a malicious actor (either through a website directly, or through the use of Man in the Middle attacks to insert scripts).

We've also looked a a wealth of other studies to find out what the state-of-the-art is in the fingerprinting of both desktop and mobile devices via both web browsers and native applications [5, 6, 26, 36, 39, 40, 44, 45, 48, 49, 51].

We've found some interesting things out too. We first asked which of mobile and desktop devices are more fingerprintable, and came to the conclusion that really, they're approximately equally fingerprintable, and quite fingerprintable at that. We saw in Figure 11 how both had approximately 90% of collected fingerprints being in their own anonymity set, and how they each had a similar proportion of other sizes of sets.

Taking the same measures, we've also found no substantial difference between iOS and Android in the analysis of the AIU Dataset. They appear to have very similar proportions of anonymity sets in the data (which reflects more recent models of device).

On otherwise identical devices (using the iPhone 7 as an example), we discovered that we could in fact distinguish between specific devices; the primary differentiator between devices was language, and then within that, time zone. However, it's unclear how one could distinguish between, for example, all British iPhone 7 users (who will share the same time zone and will likely have their locale set to en-GB or en-US). However, it's still important to note that of the 57 iPhone 7s in the AIU dataset, nearly half were uniquely identifiable, and the largest anonymity set was only 7 identically-identified phones.

We've also found that protection from fingerprinting is limited, with only some extensions blocking fingerprinting in mainstream browsers by default (uBlock Origin), and the Brave browser, which will explicitly attempt to prevent fingerprinting tools from running. However, the only browser that actually attempts to evade fingerprinting by feeding false information is the Tor Browser (on desktop), which rather than result in the discrepancy of having no data or incomplete data, will have a complete set of data. However the Tor browser is distinctive all by itself, it just isn't amongst other Tor Browser users.

We were somewhat surprised at some of these results, especially the way that configuration options between apparently identical hardware acted as a significant discriminator. We've also been a little surprised at the slightly more bizarre ways that browsers have identified themselves, such as the Amigo browser mentioned in Section 3.2.2.

It was also astonishing just how many Android browsers are identical to each other, being just a webview with a slightly different interface (though often *very* similar). It is worth noting however, that this is not the case for popular browsers, and that many of these webview-only browsers would be notable for not being stock Chrome. Having used several of them, we can attest to the fact that many of them are low-quality, with poor interfaces and frequent crashes (with some not even able to start up on our hardware).

Something especially unexpected is the fact that enabling the Do Not

Track setting will actually make a particular configuration more unique and fingerprintable, thus somewhat defeating the point of the DNT standard.

Final Thoughts: This project has given some interesting insight into the fingerprintability of various types of devices and platforms against each other.

Chapter 6

Future Work

We would like to know how else we can further extend this work and field, as this is an exciting and fairly new field with many recent developments. Here we present some thoughts about different directions we can move in:

Native app fingerprinting: We can further extend this project with work similar to that acheived by Kurtz et al. [6] in 2016, by fingerprinting with direct access to the mobile device platform APIs. It would be fascinating to run a similar project a few years down the line in order to find out what fingerprinting techniques and options are still available to us.

New HTML5 APIs: As newer HTML5 APIs are proposed, and as they come into support in mainstream browsers, more and more data, much of which could potentially be used for fingerprinting and identification purposes, will be available to web browsers on all types of devices, but especially mobile devices. We particularly would be interested to know if information about hardware components, such as batteries, networking components, and security identifiers, could be used in the future to attempt to fingerprint devices.

Execution profiles: Following on from this, and Mowery et al. [36], it would be helpful and useful for the fingerprinting community at large, in my opinion, to take and keep up to date execution profiles such as those recorded by Mowery et al., but for many different browsers running on different mobile devices, operating systems, and versions of those browsers and operating systems.

Operating Systems as a Service: It has been indicated by both Microsoft and Google that their respective flagship desktop operating systems (Windows 10 and Chrome OS) are to follow the Operating Systems as a Service model, where the system is continually updated and upgraded with new features, with little to no choice in the matter for consumers — much like web applications have been until now. Particularly as Windows 10 subsumes its primary predecessor, Windows 7, it shall be interesting to watch how this affects the fingerprintability of machines.

Fingerprinting Tor: The Tor network and its browsers and various apps make a concerted effort to hide the identity of the users, and additionally, to prevent fingerprinting, such as by blocking certain uses of HTML5 canvas. Some future work in this area may wish to more closely examine the fingerprintability and security properties of the Tor network and its various software.

Data analysis of industry data: We alluded to the limited size of the datasets that have been collected for research purposes back in Section 1.1, at least compared to the datasets that large advertising companies can gather. It would be fascinating to see how effective fingerprinting would be on such an enormous scale, and what novel techniques would be effective against such a wide cross-section of internet users. Such datasets could be from products such as Google Analytics, Google AdWords, Facebook, and Microsoft Bing Ads. However, such work would require co-operation from one of these companies, and the results may not be made available for public consumption.

Chapter 7

Evaluation

In Section 1.2, we laid out three objectives for this project and report. In this evaluation chapter, we'll try to see how well we did on each of these points.

The objectives were as follows:

1. We hope to attempt to identify novel ways of fingerprinting web browsers on mobile devices, with a focus on iOS devices (which are very similar within each model). We hope that this provides a base for further research in this area, and that it will perhaps give users of mobile devices a greater degree of privacy in their online interactions.
2. We also hope to investigate the current state of web browser fingerprinting technologies, and how they apply to mobile devices, especially iOS.
3. We hope to build on previous research in fingerprinting iOS devices such as [6], but with a web-based approach rather than the described native mobile app.

One of the biggest criticisms of this project from the author is that it cast the net a little too widely in the end. Although we had originally intended to look at just iOS devices, we actually found ourselves looking at larger selection of devices, operating systems, and web browsers, including Android, Windows Desktop, Linux Desktop, and various means of connection to the Tor network.

7.1 Fingerprinting on mobile devices

We're pleased that in mobile devices in general, analysis of existing data showed that mobile devices are as fingerprintable as desktop devices are, using the same techniques — looking at Figure 11 we can see this clearly. We've also looked at some of the more novel ways of fingerprinting mobile devices using sensors, and the ways that various academic studies have found of doing some form of fingerprinting, though these methods are not necessarily useful outside of a lab environment. We think this is now a base for further research, and it's good that we've been able to look at some more recent devices not yet shown in a study of the AIU dataset. We've also been able to do some analysis of our own and have found some specific identifying features between apparently identical models.

7.2 Current state of fingerprinting

On this criterion, we've done quite well; we've looked at a wide range of studies on fingerprinting [5, 6, 26, 39, 44, 51], on tracking [40, 45, 48, 49], and some on web advertising in general [8, 9, 11, 12, 13, 16, 17, 28, 46, 47, 50]. Much of this is referenced in Chapter 4, and Section 3.2.4, which together serve as a good summing up of the field.

7.3 iOS Device fingerprinting

As suggested above, this project has not managed to do particularly well in this objective. Most of the things we've found out as a result have been broadly applicable to mobile devices in general, rather than iOS. This is a little disappointing, but we were also prepared to find that there were no iOS-specific features that were fingerprintable, and this was suggested in the interim report's evaluation criteria, six months ago. However, we have found some things, such as hardware identifiers leaked in the Apple-mandated User-Agent string can be used to identify different devices (see Table 3). We've also found that even within identical models, in the AIU dataset, that nearly half of all of a particular model of iPhone were uniquely identifiable.

7.4 Tracking Meetings

We've met regularly with Ben, our supervisor since the end of the spring examination period in March, usually every two weeks, and slightly more often in the last month or so of the project runtime. These meetings were intended to ensure that the project is still moving forward and that it's suitably academic — indeed Ben's advice has been extremely useful in coming up with conclusions and in the presentation of results, as well as how to write a *story*, rather than just presenting some dry data. More meetings earlier on would probably have been helpful, and may have resulted in a higher quality tool more specific to mobile devices, but nonetheless we have collected most variables that can be generated by a browser without a user needing to be present (as is the case for collecting fingerprints from hardware sensors like gyroscopes).

7.5 Fingerprinting Tool

The tool that we've produced reuses existing libraries to collect data — namely fingerprintjs2 [32], though ClientJS [33] was also used to give some more refined details about fingerprints. However, no ClientJS data was actually stored in the end by the tool. As we noted in the interim report, we did not expect to collect a vast amount of data with this tool, given that to do so would require a publicity campaign *after* the tool had been produced. This isn't something that we can't rule out for the future though. The tool is basic in its presentation and design, and is not especially user-friendly (see Figure 7), but this would be fairly trivial to resolve — it just wasn't considered hugely important in the context of what we wanted to acheive with this project. We have however, been able to collect appropriate fingerprinting data in an easily analysable format, which is something we wanted to acheive. The tool doesn't push the boundaries of fingerprint collection though; it's primary purpose was and remains to enable me to collect data for the manually-curated dataset, which is a small private dataset, orders of magnitude smaller than large public projects like AmIUnique [3].

7.6 Data Analysis

A major criticism we would make of our data analysis is that it was all done manually, with SQL queries on the two datasets, or — especially in the case of the manually-curated dataset — using a spreadsheet application to compare small numbers of results. It might have been more useful, and might have made analysis quicker (enabling more to take place), to have an automated tool that would automatically generate low-level SQL queries based on higher-level ones, and upon retrieving the results, creating appropriate graphs and tables. This could be considered to be out-of-scope of this project, but it would have made the data analysis (which constitutes the substantial part of this project) easier to do, and would have produced more results either backing up or creating new insight.

7.7 Overall Evaluation

This project has met its objectives reasonably well, bar the disappointment about iOS-specific features. We're pleased to have come out with some novel and surprising results, and we're pleased to have provided some objective analysis about some security claims made by software authors, and have tested the limits of these with regards to fingerprinting (as opposed to tracking).

We've managed to add some value to existing work — so far as we know, nobody had directly compared the fingerprintability of mobile and desktop, or of the two major mobile platforms before. Of course, the results we've come up with indicate that they are broadly similar, but this is important to know too for anybody who wants to keep their communications and identity a secret.

If this project were to be repeated, we would probably focus more closely on only two or three of the research questions, perhaps the first, second, and fifth, delving deeper into those specific subjects (namely mobile compared to desktop; mobile platforms compared; and fingerprinting evasion techniques employed by browsers). Alternatively, we could focus just on research questions three and four (fingerprinting of identical models and mobile-only fingerprinting methods).

Final Thoughts This project has been enjoyable to carry out and we hope to look further into this area in the future, and at the wider context of user tracking and the security protections afforded to users of computers and different software. We hope that there will be further research in this area, focusing more closely on individual parts of the project, and believe that this project provides a good base for such further research. ■

Bibliography

[1] Gary Kovacs. *Gary Kovacs: Tracking our online trackers — TED Talk*. 2012. URL: https://www.ted.com/talks/gary_kovacs_tracking_the_trackers (visited on 15/06/2018).

[2] Electronic Frontier Foundation. *Panopticlick*. 2010. URL: https://panopticlick.eff.org/ (visited on 21/01/2018).

[3] Institut National de Recherche en Informatique et en Automatique and Institut de Recherche en Informatique et Systèmes Aléatoires. *Am I unique?* URL: https://amiunique.org/ (visited on 21/01/2018).

[4] Ian Thomson. *HTML5 may as well stand for Hey, Track Me Longtime 5. Ads can use it to fingerprint netizens*. 2018. URL: http://www.theregister.co.uk/2018/01/17/html5_online_tracking/ (visited on 21/01/2018).

[5] He Wang, Dimitrios Lymberopoulos and Jie Liu. "Sensor-based User Authentication". In: *Wireless Sensor Networks*. Cham: Springer International Publishing, 2015, pp. 168–185.

[6] Andreas Kurtz et al. "Fingerprinting Mobile Devices Using Personalized Configurations". In: *Proceedings on Privacy Enhancing Technologies* 2016.1 (2016), pp. 4–19. ISSN: 2299-0984. DOI: 10.1515/popets-2015-0027.

[7] Cindy Liu, Shelleen Shum and Martín Utreras. *WORLDWIDE AD SPENDING eMarketer's Updated Estimates and Forecast for 20152020*. 2016. URL: http://www.strathcom.com/wp-content/uploads/2016/11/eMarketer_Worldwide_Ad_Spending-eMarketers_Updated_Estimates_and_Forecast_for_20152020.pdf (visited on 22/01/2018).

[8] Enric Pujol, Oliver Hohlfeld and Anja Feldmann. "Annoyed Users: Ads and Ad-Block Usage in the Wild". In: *Proceedings of the 2015 ACM Conference on Internet Measurement Conference - IMC '15*. New York, New York, USA: ACM Press, 2015, pp. 93–106. ISBN: 9781450338486. DOI: 10.1145/2815675.2815705.

[9] Nevena Vratonjic et al. "Ad-Blocking Games: Monetizing Online Content Under the Threat of Ad Avoidance". In: *The Economics of Information Security and Privacy*. Ed. by Rainer Böhme. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 49–73. ISBN: 978-3-642-39498-0. DOI: 10.1007/978-3-642-39498-0_3.

[10] Eyeo GmbH. *About AdBlock Plus*. URL: https://adblockplus.org/about.

[11] Robert J Walls et al. "Measuring the Impact and Perception of Acceptable Advertisements". In: *Internet Measurement Conference*. ACM, 2015. DOI: 10.1145/2815675.2815703.

[12] Kiran Garimella, Orestis Kostakis and Michael Mathioudakis. "Ad-blocking: A Study on Performance, Privacy and Counter-measures". In: (2017). DOI: 10.1145/3091478.3091514.

[13] Umar Iqbal, Zubair Shafiq and Zhiyun Qian. "The Ad Wars: Retrospective Measurement and Analysis of Anti-Adblock Filter Lists". In: 13 (2017). DOI: 10.1145/3131365.3131387.

[14] Eyeo GmbH. *Acceptable Ads Manifesto*. 2014. URL: https://acceptableads.org/ (visited on 24/01/2018).

[15] Eyeo GmbH. *About — The Acceptable Ads Committee*. URL: https://acceptableads.com/en/committee/ (visited on 24/01/2018).

[16] Muhammad Haris Mughees, Zhiyun Qian and Zubair Shafiq. "Detecting Anti Ad-blockers in the Wild". In: *Privacy Enhancing Technologies*. Vol. 2017. 3, pp. 127–142. DOI: 10.1515/popets-2017-0031.

[17] Rishab Nithyanand et al. "Ad-Blocking and Counter Blocking: {A} Slice of the Arms Race". In: *CoRR* abs/1605.0 (2016). arXiv: 1605.05077.

[18] Mark Groman. *No One Should Be Outed By an Ad*. 2015. URL: https://iapp.org/news/a/nai-takes-lgbt-stand/ (visited on 24/01/2018).

[19] Adrian Chen. *How Facebook's Targeted Ads Revealed One User's Sexuality*. 2010. URL: http://gawker.com/5671582/how-facebooks-targeted-ads-revealed-one-users-sexuality (visited on 24/01/2018).

[20] Aleksandra Korolova. "Privacy violations using microtargeted ads: A case study". In: *Proceedings - IEEE International Conference on Data Mining, ICDM*. Sydney, Australia, 2010, pp. 474–482. ISBN: 9780769542577. DOI: 10.1109/ICDMW.2010.137.

[21] Digital Advertising Alliance. *YourAdChoices.com*. URL: http://youradchoices.com/ (visited on 25/01/2018).

[22] Mozilla Corporation. *Do Not Track: A Universal Third-Party Web Tracking Opt Out*. URL: https://datatracker.ietf.org/doc/draft-mayer-do-not-track/ (visited on 25/01/2018).

[23] Mark Stockley. *Do Not Track the privacy standard that's melting away Naked Security*. URL: https://nakedsecurity.sophos.com/2014/08/26/do-not-track-the-privacy-standard-thats-melting-away/ (visited on 25/01/2018).

[24] Raymond Hill. *uBlock Origin*. 2016. URL: https://github.com/gorhill/uBlock/blob/master/README.md (visited on 14/06/2018).

[25] Arthur Gervais et al. "Quantifying Web Adblocker Privacy". In: *Computer Security – ESORICS 2017*. Ed. by Simon N Foley, Dieter Gollmann and Einar Snekkenes. Cham: Springer International Publishing, 2017, pp. 21–42. ISBN: 978-3-319-66399-9.

[26] Pierre Laperdrix, Walter Rudametkin and Benoit Baudry. "Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints". In: *Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016* (2016), pp. 878–894. DOI: 10.1109/SP.2016.57.

[27] Peter Eckersley. "How unique is your web browser?" In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6205 LNCS (2010), pp. 1–18. ISSN: 03029743. DOI: 10.1007/978-3-642-14527-8_1.

[28] Grant Storey et al. "The Future of Ad Blocking: An Analytical Framework and New Techniques". In: 2017. arXiv: 1705.08568.

[29] Tamer Sameeh. *A Security Evaluation of Public and Private Tor Bridges - Deep Dot Web*. 2017. URL: https://www.deepdotweb.com/2017/02/07/a-security-evaluation-of-public-and-private-tor-bridges/ (visited on 15/06/2018).

[30] Tor Project. *Tor Project — Privacy Online*. URL: https://www.torproject.org/ (visited on 15/06/2018).

[31] BBC News. *Silk Road drug website founder Ross Ulbricht jailed - BBC News*. URL: https://www.bbc.co.uk/news/world-us-canada-32941060 (visited on 15/06/2018).

[32] Valentin Vasilyev. *fingerprintjs2*. URL: https://github.com/Valve/fingerprintjs2.

[33] *ClientJS*. URL: https://clientjs.org/ (visited on 17/06/2018).

[34] Apple. *iPhone 7 - Technical Specifications - Apple (UK)*. 2016. URL: https://www.apple.com/uk/iphone-7/specs/ (visited on 26/05/2018).

[35] Jason Packer and Quantable. *How Many of Your Users Set "Do Not Track" - Quantable.* URL: https://www.quantable.com/analytics/how-many-do-not-track/ (visited on 28/05/2018).

[36] Keaton Mowery et al. "Fingerprinting Information in JavaScript Implementations". In: *Web 2.0 Security & Privacy* (2011), pp. 1–11.

[37] Netchain. *V8 Benchmark Suite.* URL: http://www.netchain.com/Tools/v8/ (visited on 28/05/2018).

[38] Webkit. *SunSpider 1.0.2 JavaScript Benchmark.* URL: https://webkit.org/perf/sunspider/sunspider.html (visited on 28/05/2018).

[39] Hristo Bojinov et al. "Mobile Device Identification via Sensor Fingerprinting". In: (2014). arXiv: 1408.1416.

[40] Anupam Das, Nikita Borisov and Matthew Caesar. "Tracking Mobile Web Users Through Motion Sensors : Attacks and Defenses". In: February (2016), pp. 21–24.

[41] Giorgio Maone. *NoScript Security Suite – Add-ons for Firefox.* URL: https://addons.mozilla.org/en-GB/firefox/addon/noscript/.

[42] Arvind Narayanan. *The Web Tracking Arms Race: Past, Present, and Future — USENIX.* (Visited on 17/05/2018).

[43] Alexei Bulazel and Bülent Yener. "A Survey On Automated Dynamic Malware Analysis Evasion and Counter-Evasion". In: *Proceedings of the 1st Reversing and Offensive-oriented Trends Symposium on - ROOTS.* New York, New York, USA: ACM Press, 2017, pp. 1–21. ISBN: 9781450353212. DOI: 10.1145/3150376.3150378.

[44] Antoine Vastel, Walter Rudametkin and Romain Rouvoy. "FP-TESTER: Automated Testing of Browser Fingerprint Resilience". In: *Proceedings of the 4th International Workshop on Privacy Engineering (IWPE'18)* (), pp. 1–5.

[45] Yuanyi Wu, Dongyu Meng and Hao Chen. “Evaluating Private Modes in Desktop and Mobile Browsers and Their Resistance to Fingerprinting”. In: ().

[46] G Merzdovnik et al. “Block Me If You Can: A Large-Scale Study of Tracker-Blocking Tools”. In: *2017 IEEE European Symposium on Security and Privacy (EuroS P)*. 2017, pp. 319–333. DOI: 10.1109/EuroSP.2017.26.

[47] D Sculley et al. “Detecting Adversarial Advertisements in the Wild”. In: *Proceedings of the 17th ACM SIGKDD International Conference on Data Mining and Knowledge Discovery*. 2011.

[48] Sebastian Zimmeck et al. “A Privacy Analysis of Cross-device Tracking”. In: *26th {USENIX} Security Symposium ({USENIX} Security 17)* (2017), pp. 1391–1408.

[49] Elias P. Papadopoulos et al. “The Long-Standing Privacy Debate”. In: *Proceedings of the 26th International Conference on World Wide Web - WWW ’17*. New York, New York, USA: ACM Press, 2017, pp. 153–162. ISBN: 9781450349130. DOI: 10.1145/3038912.3052691.

[50] Paul Barford et al. “Adscape: Harvesting and Analyzing Online Display Ads”. In: *CoRR* abs/1407.0 (2014). arXiv: 1407.0788.

[51] Thomas Hupperich et al. “On the Robustness of Mobile Device Fingerprinting”. In: *Proceedings of the 31st Annual Computer Security Applications Conference on - ACSAC 2015*. New York, New York, USA: ACM Press, 2015, pp. 191–200. ISBN: 9781450336826. DOI: 10.1145/2818000.2818032.

Appendix A

Glossary

AdWords Google's primary advertising program, and a major source of revenue for the company.

Anonymity set An anonymity set is a set of entries within some data that cannot be distinguished from each other, excluding any ID number that might have been assigned (assuming that such an ID number is independent of the content of the rest of the data). An anonymity set of size 1 means that there is no other data that completely matches the data item inside the set, whereas an anonymity set of size 1,000 means that there are 1,000 identical pieces of data within a set. One dataset can have up to its own size in anonymity sets, with each entry being in its own anonymity set, or can have as little as one anonymity set, with all entries in the same set.

Blink The rendering and layout engine used by Chromium and its derivatives, including Google Chrome.

Chrome Both the name of a popular web browser, and the name given to the UI of a web browser around the viewport.

Cookie A small file stored on a client computer that can store information from a website. The contents of the file are sent back to the website when a webpage from that site is loaded.

Desktop A traditional personal computer system. In this study, we use desktop to refer not only to traditional "tower" PCs, but also to laptops, ultrabooks, netbooks and all-in-ones, where the monitor is also the computer system.

Discrete graphics A device separate from the CPU is responsible for most graphics. This is usually known as a graphics card.

DoC Department of Computing, Imperial College London.

DOM Document object model; the API that allows for the manipulation of HTML documents. JavaScript can usually interact with the DOM to modify the HTML of a webpage as well as to perform other functions.

Fingerprinting Gathering information from a system in an attempt to uniquely identify it. In web browsers, this is usually done by JavaScript manipulating the DOM.

Gecko Mozilla Firefox's rendering and layout engine for HTML and CSS.

Integrated graphics The graphics controller of a computer system is part of its CPU.

ISP Internet service provider; provides internet services to businesses and private individuals, usually through a street cabinet, or fibre-optic cable installed into the address where the service is required.

Man-in-the-middle The interception of a particular user's network communications, which may be modified before being sent on, either to the user, or to the other computer that they are trying to communicate with.

Mobile device A computer device that is not a desktop, nor an embedded system (like a POS), nor a server. Usually refers to smartphones and tablets, and to the hybrid of both, phablets.

Paywall Forces all users (of a website) to pay in order to access any content. An example of a site that does this is that of The Times & The Sunday Times

TLS Transport Layer Security; often (incorrectly) known as SSL, which is its predecessor, or as HTTPS. Provides encrypted connections from clients to websites using public-key cryptography.

Tor The Onion Router: so-named because traffic routed through the Tor network is surrounded by layers (of encryption). Tor is designed so that traffic cannot be traced back to its original source.

Tracking The use of an identifier, usually a cookie, to track a particular user across the web.

UUID Universally unique identifier; used to identify a particular object, which in this context, may well be a web user or their browser.

Viewport Part of the web browser that actually displays the webpage

WebKit Apple's rendering engine used in Safari and all iOS browsers, and formerly used in Chromium and its derivatives (including Google Chrome)

Webview A simple web page viewer within a mobile application. Many web browser applications on Android, and all on iOS (due to Apple mandating so), are webviews with different interfaces and with different built-in ways of controlling the DOM. A common use-case is to open a company's support forum or terms of service page, rather than showing them within the confines of the app. Webviews can often be "broken out" of resulting in access to the entire web within an app not necessarily designed to do so — this can often be done on the BBC News app, for example.

Windows NT Often now referred to just as Windows, or Microsoft Windows, is the name both of the old Windows NT brand, and of newer business operating systems like Windows 2000, and of the consumer and business operating system produced since Windows XP and through to Windows 10.