HAL
open science

Beauty and the Beast: Diverting modern web browsers to build unique browser fingerprints

Pierre Laperdrix, Walter Rudametkin, Benoit Baudry

To cite this version:

Pierre Laperdrix, Walter Rudametkin, Benoit Baudry. Beauty and the Beast: Diverting modern web browsers to build unique browser fingerprints. 37th IEEE Symposium on Security and Privacy (S&P 2016), May 2016, San Jose, United States. hal-01285470v1

HAL Id: hal-01285470

https://inria.hal.science/hal-01285470v1

Submitted on 9 Mar 2016 (v1), last revised 14 Mar 2016 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Beauty and the Beast: Diverting modern web browsers to build unique browser fingerprints

Pierre Laperdrix
INSA-Rennes & INRIA
Rennes, France
pierre.laperdrix@insa-rennes.fr

Walter Rudametkin
University of Lille & INRIA
Lille, France
walter.rudametkin@univ-lille1.fr

Benoit Baudry
INRIA
Rennes, France
benoit.baudry@inria.fr

***Abstract*—Worldwide, the number of people and the time spent browsing the web keeps increasing. Accordingly, the technologies to enrich the user experience are evolving at an amazing pace. Many of these evolutions provide for a more interactive web (e.g., boom of JavaScript libraries, weekly innovations in HTML5), a more available web (e.g., explosion of mobile devices), a more secure web (e.g., Flash is disappearing, NPAPI plugins are being deprecated), and a more private web (e.g., increased legislation against cookies, huge success of extensions such as Ghostery and AdBlock).**

Nevertheless, modern browser technologies, which provide the beauty and power of the web, also provide a darker side, a rich ecosystem of exploitable data that can be used to build unique browser fingerprints.

Our work explores the validity of browser fingerprinting in today's environment. Over the past year, we have collected 118,934 fingerprints composed of 17 attributes gathered thanks to the most recent web technologies. We show that innovations in HTML5 provide access to highly discriminating attributes, notably with the use of the Canvas API which relies on multiple layers of the user's system. In addition, we show that browser fingerprinting is as effective on mobile devices as it is on desktops and laptops, albeit for radically different reasons due to their more constrained hardware and software environments. We also evaluate how browser fingerprinting could stop being a threat to user privacy if some technological evolutions continue (e.g., disappearance of plugins) or are embraced by browser vendors (e.g., standard HTTP headers).

***Index Terms*—browser fingerprinting; privacy; software diversity**

I. Introduction

The world wide web has revolutionized communication in just a few decades. The number of users and the time spent on the web is constantly growing. Accordingly, the technologies to enrich the user experience are evolving at an amazing pace. Each technology has its purpose. Modern Javascript libraries allow creating ever more dynamic and interactive web applications. Users are bringing the web with them, wherever they go, by means of mobile devices such as cellphones and tablets. Browser and protocol specifications, such as HTML5, are redefining the limits of what web applications can do. The browsers themselves are rapidly changing and have become competitive testing grounds for numerous new technologies. Surprisingly, what were once ubiquitous technologies, such as the Flash, Silverlight, QuickTime, and Java plugins, are quickly becoming relics of the past. At the same time, concerned web user's are becoming more aware of certain practices that jeopardize their privacy and comfort, as can be seen by the immense popularity of browser extensions like AdBlock, Ghostery, Disconnect and many others.

Browsers are our gateway to the web. And to provide rich, satisfying and beautiful services, websites require knowledge about the browser and its environment. Through the different APIs and technologies that have been created, modern browsers freely provide websites with detailed information regarding the hardware and software configuration, allowing websites to better exploit the user's resources. Well behaving websites only ask for what is needed to provide their beautiful services, but the beast is hiding in the bushes, small differences between users' systems can be exploited by attackers by asking for as much information as possible.

Browser fingerprinting consists in collecting data regarding the configuration of a user's browser and system when this user visits a website. This process can reveal a surprising amount of information about a user's software and hardware environment, and can ultimately be used to construct a unique identifier, called a *browser fingerprint*. The privacy implications are important because these fingerprints can then be used to track users. This threat to privacy is extremely serious as assessed by the recent studies of Nikiforakis et al. [1] or of Acar et al. [2] that show the wide adoption of browser fingerprinting. Meanwhile, large companies such as Google implicitly announce its adoption (e.g., Google's privacy policy update of June 2015 indicates that they use "technologies to identify your browser or device" [3], which can be interpreted as the inclusion of browser fingerprinting in their identification technologies).

Our work provides an in-depth analysis of the extent to which today's web provides an effective means to uniquely identify users through browser fingerprinting. This analysis relies on more than 118,000 fingerprints, which we collected through the AmIUnique.org website. The fingerprints are rich and include the values of 17 attributes. We access some of these attributes thanks to the most recent web technologies, such as, the HTML5 canvas element (as initially suggested by Mowery and colleagues [4]), as well as through the WebGL API. These fingerprints reveal detailed information about a browser and its software and hardware environment. We show that innovations in HTML5 provide access to highly

discriminating data. In addition, we provide the first extensive study about browser fingerprinting on mobile devices, which are quickly becoming the main platform for browsing the web [5]. Through empirical evidence, we show that browser fingerprinting is effective on mobile devices despite having software environments that are much more constrained than on desktops and laptops. In fact, the discriminating attributes for mobile devices differ greatly from their desktop and laptop counterparts.

Our empirical observations indicate that, while recent web technologies enrich the user experience, they also provide access to a wide range of information that are easily combined into a fingerprint that is most likely unique. The tension between the comfort of web browsing and the will to remain anonymous is currently clearly in favor of comfort, to the detriment of privacy. Yet, the disappearance of severely discriminating attributes on desktops (e.g. obtained through Flash), and the absence of such attributes on mobile devices, allows us to believe it is possible to improve privacy and anonymity on the web while still retaining a modern and comfortable web experience. We speculate on possible technological evolutions in web browsers and we calculate their impact on browser fingerprinting. Our scenarios range from the definitive death of Flash (49% of the visitors on AmIUnique.org had Flash disabled), to the premature disappearance of JavaScript . We show that minor changes in web technologies would have a major effect on the identification capacity of browser fingerprinting.

Our key contributions are:

- We provide a 17-attribute fingerprinting script that uses modern web technologies.
- We perform the first large-scale study of Canvas fingerprinting by following a test reported by Acar et al. [6] along with other JavaScript attributes. We show that canvas fingerprinting is one of the most discriminating attributes.
- We demonstrate the effectiveness of mobile device fingerprinting with 81% of unique mobile fingerprints in our dataset despite the lack of plugins and fonts. We show that the wealth of mobile models (different vendors with different firmware versions) result in very rich user-agents and very revealing canvas usage.
- We explore scenarios of possible technological evolutions to improve privacy, and we simulate their impact on browser fingerprinting using our dataset. Notably, we find out that removing plugins and having generic HTTP headers could reduce desktop fingerprint's uniqueness by a very strong 36%.

The paper is organized as follows. Section II describes our script and provides descriptive statistics about our dataset. Section III investigates the impact of the most recent technology on fingerprint diversity and section IV details the analysis of mobile fingerprint diversity. Section V evaluates the impact of possible future scenarios on fingerprint-based identification, section VI discusses the related work while section VII concludes this paper.

II. Dataset

We launched the AmIUnique.org website in November 2014 to collect browser fingerprints with the aim of performing an in-depth analysis of their diversity. The first part of this section presents the set of attributes that we collect in our browser fingerprinting script and the technique we use to collect them. Then, we give a few general descriptive statistics about the 118,934 fingerprints that serve as our dataset. We finish this section with a series of tests to compare our dataset with the only other available set of fingerprint statistics, provided by Eckersley in 2010 [7].

A. AmIUnique.org

1) Fingerprinting script: We implemented a browser fingerprinting script that exploits state-of-the-art techniques [4], [6] as well as some new browser APIs. The complete list of attributes is given in the 'Attribute' column of Table I. The 'Source' column indicates the origin of each attribute (HTTP, JavaScript or Flash). The 'Distinct values' and 'Unique values' columns give a global overview of the most discriminating attributes in a fingerprint. Finally, the last column displays a complete example of a browser fingerprint. The top 10 attributes have been presented by Eckersley. Most of the 7 attributes at the bottom of the table have been discussed in other works. Yet, we are the first to collect them on a large scale basis and to combine them as part of a fingerprint. We detail these 7 attributes below

- List of HTTP headers: When connecting to a server, browsers send the user-agent, the desired language for a webpage, the type of encoding supported by the browser, among other headers. Some software and browser extensions modify or add headers, giving extra details about the device's configuration. Being defined in the HTTP protocol, these headers can always be acquired by the server and do not depend on JavaScript.
- Platform: The value in the "*navigator.platform*" property provides information about the user's operating system. While this information is already in the user-agent, we collect the 'platform' value to detect modified or inconsistent fingerprints, e.g., in case the returned value is different from the one in the user-agent.
- Do Not Track/Use of an ad blocker: These two attributes are directly related to privacy and the values can help us differentiate privacy-conscious users from others.
- WebGL Vendor and Renderer: Described by Mowery et al. [4], these two attributes were added with the HTML WebGL API to give information on the underlying GPU of the device. We provide extensive details about the contents of these attributes in section III.
- Canvas: Introduced by Acar et al. [6] and fully explained in section III-A, the HTML5 Canvas element gives us the ability to perform tests on both the hardware and the operating system by asking the browser to render a picture following a fixed set of instructions.

TABLE I
BROWSER MEASUREMENTS OF AMIUNIQUE FINGERPRINTS WITH AN EXAMPLE

Attribute	**Source**	**Distinct values**	**Unique values**	**Example**
User agent	HTTP header	11,237	6,559	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2272.118 Safari/537.36
Accept	HTTP header	131	62	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Content encoding	HTTP header	42	11	gzip, deflate, sdch
Content language	HTTP header	4,694	2,887	en-us,en;q=0.5
List of plugins	JavaScript	47,057	39,797	Plugin 1: Chrome PDF Viewer. Plugin 2: Chrome Remote Desktop Viewer. Plugin 3: Native Client. Plugin 4: Shockwave Flash...
Cookies enabled	JavaScript	2	0	yes
Use of local/session storage	JavaScript	2	0	yes
Timezone	JavaScript	55	6	-60 (UTC+1)
Screen resolution and color depth	JavaScript	2,689	1,666	1920x1200x24
List of fonts	Flash plugin	36,202	31,007	Abyssinica SIL,Aharoni CLM,AR PL UMing CN,AR PL UMing HK,AR PL UMing TW...
List of HTTP headers	HTTP headers	1,182	525	Referer X-Forwarded-For Connection Accept Cookie Accept-Language Accept-Encoding User-Agent Host
Platform	JavaScript	187	99	Linux x86_64
Do Not Track	JavaScript	7	0	yes
Canvas	JavaScript	8,375	5,533	Cwm fjordbank glyphs vext quiz, 😃 Cwm fjordbank glyphs vext quiz, 😃
WebGL Vendor	JavaScript	26	2	NVIDIA Corporation
WebGL Renderer	JavaScript	1,732	649	GeForce GTX 650 Ti/PCIe/SSE2
Use of an ad blocker	JavaScript	2	0	no

It should be noted that the WebGL Vendor and WebGL Renderer attributes were added after our site was launched. We isolated the results obtained from these two attributes (values collected after fingerprint number 45,474).

We tested other attributes for inclusion in the fingerprints, but the results were inconclusive and we decided to discard them. We designed a test that renders 3D volumes through the WebGL API, as first tested by Mowery et al. [4]. However, after an early analysis of more than 40,000 fingerprints, the test proved to be too brittle and unreliable since a simple page reload with a different window size on a single device could change the value of this test. Appendix B goes into more details on this WebGL test. We also tested the collection of information based on the device's hardware performance, like the Octane JavaScript benchmark, but they proved to be too long and too intensive to execute. Finally, we included other Flash attributes that proved to be useful to detect inconsistencies, but did not increase fingerprint uniqueness. More details can be found in Appendix C.

2) Data collection: AmIUnique.org is a website dedicated to browser fingerprinting, aimed both at collecting data about device diversity and at informing users about the privacy implications of fingerprinting. All visitors are informed of our goal with links to both our privacy policy and FAQ sections, and they have to explicitly click on a button to trigger the collection of their device's fingerprint.

When the user initiates the connection to the page that contains our fingerprinting script, the server immediately collects the HTTP headers. Then, if the user has not blocked JavaScript, the browser runs the script that collects the bulk of the fingerprint data. If Flash is present, we go one step further and collect additional data. Our script takes a few hundred milliseconds to create a fingerprint. The contents of each fingerprint is dependent on the browser, its configuration, and the hardware and software environment it is running in.

We distinguish three main categories of fingerprints in our dataset: those with JavaScript and Flash activated (43% of the fingerprints), those with JavaScript activated but not Flash (41%), and those with no JavaScript, and hence, no Flash (16%). Given that our work focuses on fingerprinting modern browsers and at analyzing the importance of the attributes in Table I, we do not consider fingerprints with no JavaScript. Fingerprints without JavaScript only include values for the HTTP headers (i.e., 5 attributes), which drastically removes most of the functionality we are studying.

To prevent collecting multiple copies of the same fingerprint from the same user, we store a cookie on the user's device with a unique ID, and we also keep a hashed version of the IP

address. These two pieces of information allow us to identify returning devices, which represent a negligible part of our dataset.

We communicated our website on Slashdot, Framasoft, Clubic, social media channels like Facebook and Twitter, and newspapers like Le Monde. As of February 15th, 2015, we collected 142,023 fingerprints, which were then reduced to 118,934 once we removed the fingerprints without JavaScript for this study. However, because our website focuses on a very specific subject, our visitors are likely saavy Internet users who are aware of potential online privacy issues. Hence, our data is biased towards users who care about privacy and their digital footprint, and their devices might have fingerprints different than those we could collect from a more general audience.

B. Descriptive statistics

Tables I and II summarize the essential descriptive statistics of the AmIUnique dataset. Table II presents the distribution of plugins, fonts and headers in our dataset. To obtain these numbers, we decomposed each list of values into single elements and we studied how common they are by looking at the number of fingerprints in which each element is present. We divided the results from the plugins, fonts and headers into three categories: the ones that belong to less than 1% of collected fingerprints, the ones present in less than 0,1% of fingerprints, and the ones that appear in only one or two fingerprints.

Unique and distinct values: The 'Distinct values' column in Table I provides the number of different values that we observed for each attribute, while the 'Unique values' column provides the number of values that occurred a single time in our dataset. For example, attributes like the use of cookies or session storage have no unique values since they are limited to "yes" and "no". Other attributes can virtually take an infinite number of values. For example, we observed 6,559 unique values for the user-agent attribute. This is due to the many possible combinations between the browser, its version and the operating system of the device. It is extremely likely that visitors who use an exotic OS with a custom browser, such as Pale Moon on Arch Linux, will present a very rare user-agent, thus increasing the likelihood of being identified with just the user-agent.

These numbers show that some attributes are more discriminating than others, but they all contribute to building a unique and coherent fingerprint.

Plugins: We observed 2,458 distinct plugins, assembled in 47,057 different lists of plugins. They cover an extremely wide range of activities, as for example, reading an uncommon file format in the browser (e.g. FLAC files with the VLC Browser plugin), communicating with an antivirus or a download client, launching a video game directly in the browser, site-specific plugins for added functionality, etc. Some plugins are so specific that they leak information beyond the computer, like the company the user works for or the brand of smartphone, camera or printer he or she uses. 97% of plugins appear in less than 1% of collected fingerprints and 89% in less then 0,1%. A lot of plugins are created for precise and narrow uses allowing their users to be easily identified.

TABLE II
SUMMARY OF STATISTICS

Attr.	Total	<1% FP	<0,1% FP	< 3 FP
Plugin	2,458	2,383 (97%)	2,195 (89%)	950 (39%)
Font	223,498	221,804 (99%)	217,568 (97%)	135,468 (61%)
Header	222	205 (92%)	182 (82%)	92 (41%)

Fonts: We observed 221,804 different fonts, assembled in 36,202 different lists of fonts. This really high number shows the incredible wealth that exists: fonts for support of an additional alphabet, fonts for web designers, fonts for drawing shapes and forms, fonts for different languages, etc. On average, a Windows or Mac user has two to three times the amount of fonts of a Linux user. Also, 97% of fonts appear in less than 0,1% of fingerprints and a little less than 2/3 of them are only in one or two fingerprints. These percentages show how efficient a list of fonts can be for fingerprinting and transitively how critical it can be for users who want to protect their privacy. However, this list is provided through the Flash plugin, which is progressively disappearing from the web. We will see in section V that removing access to the list of fonts has a small impact on identification.

HTTP headers: We observed 222 different HTTP headers, assembled in 1,182 different lists of headers. New headers are added to the standardized ones for different reasons and from different sources. Some examples include the following:

- The browser. For example, the Opera browser on smartphones adds a *X-OperaMin-Phone-UA* header, and the Puffin browser adds a *X-Puffin-UA* header.
- A browser extension. For example, the FirePHP extension for Firefox adds the *x-FirePHP* and the *x-FirePHP-Version* headers to each HTTP request.
- The network on which you are connected. Some headers show the use of proxies or protection systems.

As indicated in Table II, 182 headers out of 222 appear in less than 0,1% of the collected fingerprints, and 92 of them come from only one or two fingerprints. These statistics mean that some HTTP headers are highly discriminating and their presence greatly affects the uniqueness of one's fingerprint.

C. Statistical validity of the dataset

This section presents a series of tests to compare our dataset with the fingerprinting statistics provided by Eckersley in 2010.

1) Mathematical treatment:

Entropy: We use entropy to quantify the level of identifying information in a fingerprint. The higher the entropy is, the more unique and identifiable a fingerprint will be.

Let H be the entropy, X a discrete random variable with possible values $\{x_1, ..., x_n\}$ and $P(X)$ a probability mass function. The entropy follows this formula:

$$H(X) = -\sum_i P(x_i) \log_b P(x_i)$$

TABLE III
NORMALIZED ENTROPY FOR SIX ATTRIBUTES COLLECTED BOTH BY PANOPTICLICK AND AMIUNIQUE

Attribute	AmIUnique	Panopticlick
User agent	0.570	0.531
List of plugins	0.578	0.817
List of fonts	0.446	0.738
Screen resolution	0.277	0.256
Timezone	0.201	0.161
Cookies enabled	0.042	0.019

We use the entropy of Shannon where $b = 2$ and the result is in bits. One bit of entropy reduces by half the probability of an event occurring.

Normalized Shannon's entropy: To compare both the AmIUnique and Panopticlick datasets, which are of different sizes, we use a normalized version of Shannon's entropy:

$$\frac{H(X)}{H_M}$$

H_M represents the worst case scenario where the entropy is maximum and all values of an attribute are unique ($H_M = \log_2(N)$ with N being the number of fingerprints in our dataset).

The advantage of this measure is that it does not depend on the size of the anonymity set but on the distribution of probabilities. We are quantifying the quality of our dataset with respect to an attribute's uniqueness independently from the number of fingerprints in our database. This way, we can qualitatively compare the two datasets despite their different sizes.

2) Comparison with Panopticlick:

Entropy: Table III lists the normalized Shannon's entropy for six different attributes for both the AmIUnique and the Panopticlick datasets. For fairness of comparison, we used our dataset in its entirety by keeping fingerprints without JavaScript. We observe that the entropy values for both datasets are similar for all attributes except for the list of plugins and the list of fonts.

For the list of plugins, it is still the most discriminating attribute but a difference of 0.24 is present. It can be explained by the absence of plugins on mobile devices which are increasingly used to browse the web and by the lack of support for the old NPAPI plugin architecture on Chrome since April 2015 (more details in section V).

For the list of fonts, a noticeable drop of 0.29 occurs because half of the fingerprints in the AmIUnique dataset were collected on browsers that do not have the Flash plugin installed or activated. Since our fingerprinting script collects the list of fonts through the Flash API, this means half of our fingerprints do not contain a list of fonts, reducing its entropy. The absence of Flash can be explained (i) by the lack of Flash on mobile devices; (ii) by the fact that the visitors of AmIUnique are privacy conscious and tend to deactivate Flash. Yet, we notice that the entropy of the list of fonts is still high.

The small value of entropy for the timezone shows that our dataset is biased towards visitors living in the same geographical areas. A higher level of entropy would have meant a more spread distribution of fingerprints across the globe.

Distribution of fingerprints: We compared frequency distributions w.r.t. anonymity set sizes from both datasets and observed very similar trends. We also studied each attribute separately and observed that the most discriminating attributes are still the ones found by Eckersley with the addition of new efficient techniques like canvas fingerprinting. More details on the distributions can be found in Appendix D.

III. FINGERPRINTING WITH THE MOST RECENT WEB TECHNOLOGIES

AmIUnique collects 17 attributes to form a browser fingerprint. Out of the 118,934 fingerprints that we study, 89.4% are unique. In this section, we analyze how the attributes collected with the most recent technologies (7 attributes at the bottom of Table I) contribute to the uniqueness of fingerprints.

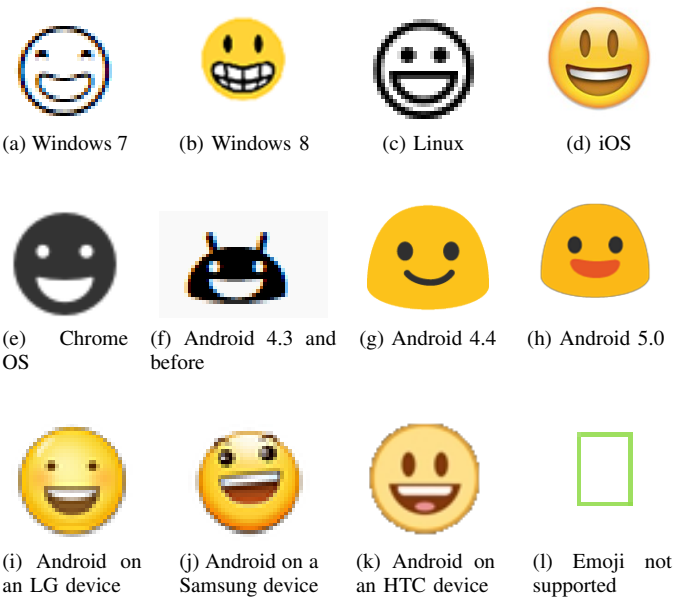
A. Canvas fingerprinting

The canvas element in HTML5 [8] allows for scriptable rendering of 2D shapes and texts. This way any website can draw and animate scenes to offer visitors dynamic and interactive content. As discovered by Mowery and al. [4] and investigated by Acar and al. [6], canvas fingerprinting can be used to differentiate devices with pixel precision by rendering a specific picture following a fixed set of instructions. This technique is gaining popularity in tracking scripts due to the fact that the rendered picture depends on several layers of the system (at least the browser, OS, graphics drivers and hardware).

1) Our test: The fingerprinting script used by AmIUnique includes a test based on the canvas element. With this image, we collect information about three different attributes of the host device, as discussed below.

Figure 1 displays the image that we use, as it is rendered by a Firefox browser running on Fedora 21 with an Intel i7-4600U processor. Our test replicates the test performed by AddThis and described in details by Acar et al [6]: print a pangram twice with different fonts and colors, the U+1F603 unicode character and rectangle with a specific color. The only adaptation is to change the position of the second string so that it is not intertwined with the first one. More details about this test are discussed below.

Fig. 1. Example of a rendered picture following the canvas fingerprinting test instructions



(a) Windows 7 (b) Windows 8 (c) Linux (d) iOS

(e) Chrome OS (f) Android 4.3 and before (g) Android 4.4 (h) Android 5.0

(i) Android on an LG device (j) Android on a Samsung device (k) Android on an HTC device (l) Emoji not supported

Fig. 2. Comparison of the "Smiling face with open mouth" emoji on different devices and operating systems

Font probing: This test captures OS diversity. The script tells the browser to render the same *pangram* (a string with all the letters of the alphabet) twice. For the first line we force the browser to use one of its fallback fonts by asking for a font with a fake name. Depending on the OS and fonts installed on the device, the fallback font differs. For the second line the browser is asked to use the **Arial** font that is common in many operating systems and is used for the hardware and OS fingerprinting described next.

Device and OS fingerprinting: The last character of our string may be the most important one. This character should not be confused with an emoticon, which is a succession of letters, numbers and punctuation marks like ":)" or "<3" to describe an emotion. The character is an **emoji** [9]. Officially introduced in the Unicode standard 6.0 in 2010, emojis are ideograms that represent emotions or activities. The difference with emoticons is that emojis have their own Unicode character and font developers must provide their own implementation for a given emoji w.r.t. its description. Consequently, emojis can be used for fingerprinting because their actual representation differs between systems.

Figure 2 shows representations of the "Smiling face with open mouth" emoji on different operating systems and mobile devices. A square means that the browser has not found a single font on the device that supports that emoji. The use of emojis can be a powerful technique to uncover information, especially on mobile devices where phone manufacturers provide their own sets of emojis.

Hardware and OS fingerprinting: As demonstrated by Mowery et al. [4], small pixel-level differences can be detected between browsers when rendering images, even on the same OS and browser. The second line of text of the canvas test uses the Arial font. Although this font has the same dimensions across operating systems, there are visible variations of pixels in the final image due to differences in the rendering process. The process to render an image is complex and depends on both hardware and software (e.g. GPU, rendering engine, graphic drivers, anti-aliasing, OS), and this test is affected by variations in any of these layers. Interestingly, the test is also relatively stable over time because users do not often change the configuration of layers in the rendering process.

2) Influence of canvas fingerprinting for identification: The strength of canvas fingerprinting comes from the fact that it combines the three tests listed before. Alone, as a simple rendered picture, the normalized entropy is at 0.491, putting it in the top 5 of the most discriminating attributes. However, because emojis reveal information about both the OS and the device, it is possible to use canvas fingerprinting to detect inconsistent fingerprints. For example, by checking if the operating system in the user-agent matches the one indicated by the emoji, we can verify inconsistencies in the fingerprint to detect visitors who spoof their fingerprintable attributes. Thus, *the added value of canvas fingerprinting is to strengthen the identity of a fingerprint*. Moreover, one of the advantages of canvas fingerprinting is that it is stable. You can run it many times on the same computer and you will have the same result every time, with little variance over time (some variations can be observed if the user decides to update drivers for example). In the end, canvas fingerprinting is an important addition to browser fingerprinting.

B. WebGL fingerprinting

WebGL [10] uses the Canvas element described before to render interactive 3D objects natively in the browser, without the use of plugins. With the final specifications in 2011, WebGL 1.0 is now supported in all major browsers.

1) Our test: The WebGL API, through the *WEBGL_debug_renderer_info* interface (as the name indicates, it is designed for debugging purposes), gives access to two attributes that take their values directly from the device's underlying graphics driver. AmIUnique's fingerprinting script collects these two properties, namely:

- the WebGL vendor: name of the vendor of the GPU.
- the WebGL renderer: name of the model of the GPU.

These attributes provide very precise information about the device. For example, we collected exact GPU names like "NVIDIA GeForce GTX 660 Ti" or "Intel HD Graphics 3000". These two attributes also indirectly leak information on your OS and its environment. For example, Chrome uses the ANGLE backend [11] on Windows to translate OpenGL API calls to DirectX API calls. Consequently, the following WebGL renderer string indicates that the browser runs on a Windows machine: "ANGLE (NVIDIA GeForce GTX 760 Direct3D11 vs_5_0 ps_5_0)". Same type of leak with the presence of the "OpenGL engine" substring on Mac systems.

2) Influence of WebGL fingerprinting on identification: The WebGL vendor and renderer had the potential to become a highly discriminating attribute, but two factors greatly hamper

its utility. First, not all browsers give the unmasked version of the vendor and renderer. Chrome provides this information by default but Firefox has this information locked behind a browser flag ("webgl.enable-privileged-extensions") and returns a simple "Not supported" with our script. Second, a non-negligible number of devices share the same hardware. For example, a lot of laptops do not have a dedicated GPU and they use the embedded Intel GPU inside their processor. This reduces the uniqueness of some of the values that we can observe. In the end, the *WebGL API opens the door to discriminating information but it is not accessible from every browser*.

C. Additional attributes

We collected the following attributes to study their utility to discriminate browsers, to strengthen a fingerprint by verifying values, and to detect inconsistencies.

Platform: Even though the platform attribute does not add new information, it can be used to detect inconsistencies. For example, on an unmodified device, if the browser indicates in its user-agent that it is running on a Linux system, you expect to see "Linux" as the value of the "*platform*" property. Due to the nature of our website that incites users to modify their browser, we flagged 5,426 fingerprints in our dataset as being inconsistent. Some browsers gave completely random values that had no meaning. Others used extensions to mask the platform value. For example, one fingerprint had the value "masking-agent", indicating that the Masking Agent extension for Firefox [12] was installed. Finally, other browsers modified their user-agent to mimic one from another operating system. The problem was that the platform property was not modified and the script was able to identify the true operating system that the user was trying to hide.

Even with its low entropy, the *platform* property can prove useful in cases where it is badly modified because it can make some devices more prone to identification than others with unique or unusual values.

Do Not Track & Ad blocker: These two attributes have a very low-level of entropy, their values are either 'Yes", "No" or "Not communicated" (for the DNT preference). Without the Do Not Track attribute, the percentage of unique fingerprints drops by 0.07% which is negligible. The Ad Blocker attribute is slightly better, with a drop of 0.5%, but still insignificant compared to other attributes like the user-agent or the list of plugins.

To conclude this section, the additional attributes collected by AmIUnique are game changers: they strengthen fingerprints, allow identification through inconsistency detection. They also allow identification even when the list of fonts is inaccessible because of the absence of Flash, and they provide essential information about browsers on mobile devices as it will be detailed in the next section.

IV. Mobile fingerprint diversity

Given the growth of mobile devices to browse the web, it is essential to analyze how browser fingerprinting behaves in this context. Our analysis of mobile device fingerprinting is based on 13,105 mobile fingerprints. We select these fingerprints from our dataset by analyzing the user-agents. If the user-agent contains a substring that is present in a predefined set ('Mobile', 'Android', 'iPhone' or 'iPad'), the fingerprint is selected as a mobile fingerprint, otherwise, it belongs to the desktop/laptop category.

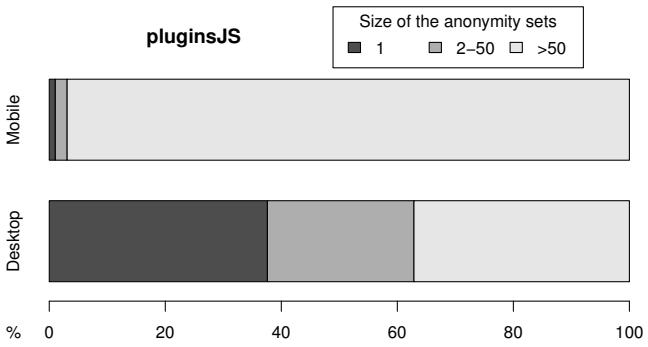


Fig. 3. Comparison of anonymity set sizes on the list of plugins between desktop and mobile devices

In this section, we first compare desktop/laptop fingerprints with mobile ones. Then, we perform a detailed analysis of mobile fingerprints, looking at differences between browsers and between mobile operating systems.

A. Mobile and Desktop fingerprint comparison

Using the attributes from Table I, we succeeded in uniquely identifying 90% of desktop fingerprints. This number is lower for mobile fingerprints at 81%, yet still quite effective. At first sight, the overall results are close. However, as we discuss in this section, the discriminating attributes for mobile fingerprints are very different from those for desktop fingerprints. One factor is the lack of plugins in general, and Flash in particular, for mobile devices. We also discuss the importance of the new attributes collected through the HTML5 canvas and WebGL elements on mobile device fingerprinting.

If we take a look at Figure 3, we can clearly notice an important difference. For desktops, more than 37% of the collected fingerprints have a unique list of plugins, while it is at 1% for mobile devices. This is due to the fact that mobiles were designed to take full advantage of HTML5 functionalities and do not rely on plugins. For example, Adobe removed the Flash player from the Google Play store in August 2012 as part of a change of focus for the company [13]. Plugins are considered to be unsuitable for the modern web and Google states in their move to deprecate NPAPI support for their Chrome browser that these plugins are a source of " hangs, crashes, security incidents, and code complexity" [14]. This choice helps mobile device users gain some privacy with regards to fingerprint uniqueness. The level of entropy of the plugin attribute is close to zero (some iOS systems have the QuickTime plugin and some Android systems reported having Flash, possibly from legacy installations). The lack of plugins also reduces information leaks that could come from them. In

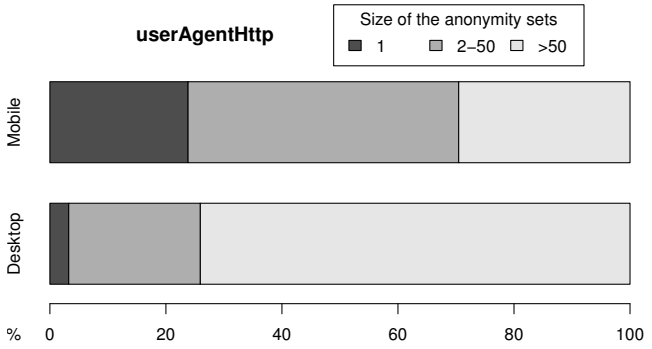


Fig. 4. Comparison of anonymity set sizes on the user-agent between desktop and mobile devices

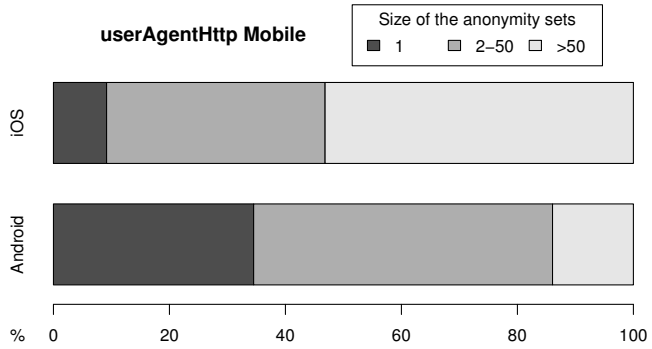


Fig. 5. Comparison of anonymity set sizes on the user-agent between Android and iOS devices

particular, mobile phones and tablets do not have the Flash plugin, thus all the fingerprint attributes leaked through the Flash API are unavailable.

Despite the unavailability of the two most discriminating attributes from desktop fingerprints (list of fonts and plugins), mobile fingerprints are still very much recognizable. This is due to two main factors: **very rich and revealing user agents and very discriminating emojis**.

Figure 4 shows that user-agents found on mobiles are five times more unique than the ones found on desktops. In our dataset, about 1 smartphone out of 4 is instantaneously recognizable with just the user-agent. This is due to two factors:

- Phone manufacturers include the model of their phone and even the version of the Android firmware directly in the user-agent.
  Example:

  ```
  Mozilla/5.0 (Linux; Android 5.0.1;
      Nexus 5 Build/LRX22C) AppleWebKit
      /537.36 (KHTML, like Gecko) Chrome
      /40.0.2214.109 Mobile Safari/537.36
  ```

- On a smartphone, applications are slowly replacing the default browser and they have access to a wide range of personal information after the user has explicitly granted specific permissions. The problem is any of these information can be exposed for the world to see by the application. We noticed in our dataset that a lot of user-agents collected on mobile devices were sent by an application and not by the native browser.
  Example with the Facebook app where the phone carrier (Vodafone UK) and the exact model of the phone (“iPhone7" = iPhone 6 Plus) is included in the user-agent:

  ```
  Mozilla/5.0 (iPhone; CPU iPhone OS 8
      _1_1 like Mac OS X) AppleWebKit
      /600.1.4 (KHTML, like Gecko) Mobile
      /12B436 [FBAN/FBIOS;FBAV
      /20.1.0.15.10;FBBV/5758778;FBDV/
      iPhone7,2;FBMD/iPhone;FBSN/iPhone
      OS;FBSV/8.1.1;FBSS/2; FBCR/
      vodafoneUK;FBID/phone;FBLC/en_GB;
      FBOP/5]
  ```

  Sometimes, even the model of the phone can give away your phone carrier. One fingerprint reported “SM-G900P". It is a Samsung Galaxy S5 and the “P" is unique to the Sprint phone carrier.

The second highest source of entropy for mobile devices comes from canvas fingerprinting. Mobiles have unique hardware impacting the final rendered picture as explained in section III-A and emojis can also be really discriminating between two devices. As seen in Figure 2, some manufacturers have their own set of emojis and even between different versions of Android, the emojis have evolved, splitting the Android user base into recognizable groups.

In the end, desktop and mobile fingerprints are somehow equally unique in the eyes of browser fingerprinting even though the discriminating information does not come from the same attributes.

The complete details of attributes’ entropy between desktop and mobile devices can be found in Table A of the Appendix.

B. Comparison Mobile OS and browsers

More than 97% of mobile fingerprints collected on AmIUnique are either running Android or iOS: 7,416 run on Android and 5,335 on iOS. How diverse is the set of fingerprints coming from both of these operating systems?

Figure 5 shows the size of anonymity sets for user-agents on both Android and iOS devices. We can see that user agents on Android devices expose more diversity with three times as many users being in an anonymity set of size 1 (9% for iOS devices and 35% for Android devices). This is due to the wealth of Android models available on the market. Moreover, our dataset may not be representative enough of the global diversity of Android devices so these percentages may be even higher in reality. For iOS devices, the diversity is still high but much less pronounced since users share devices with identical configurations. We can notice a trend where half of the collected iOS fingerprints are in really large anonymity sets. The fact that Apple is the only manufacturer of iOS devices shows in this graph.

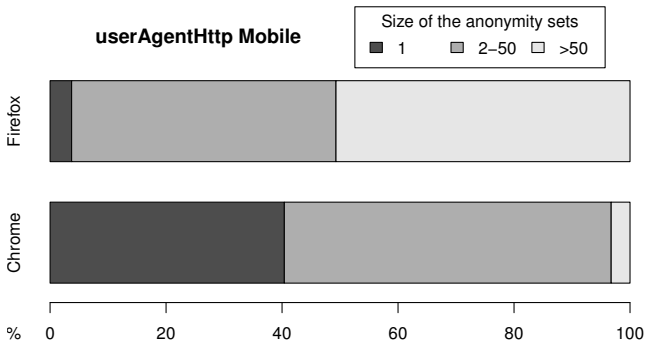


Fig. 6. Comparison of anonymity set sizes on the user-agent between Chrome and Firefox on mobile devices

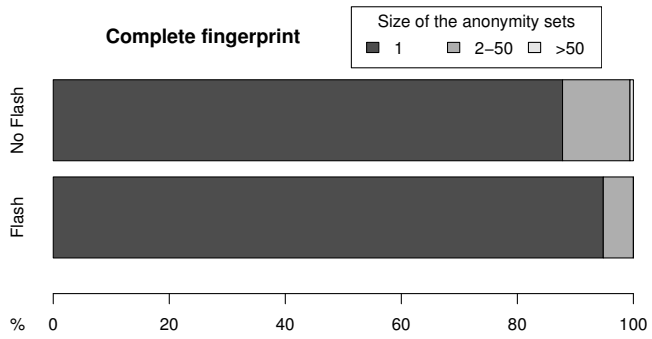


Fig. 7. Comparison of anonymity set sizes between devices with and without Flash

We saw in the previous section that user-agents can give really discriminating information on the user's device. Some smarpthones running Android give the exact model and firmware version of their phone. Looking at Figure 6, user agents from the Chrome mobile browser are ten times more unique than user agents from the Firefox browser (40% against less than 4%). This can be explained by the fact that the Chrome browser is the default browser on Android and it is automatically installed on every devices. When a phone manufacturer builds its tailored firmware to be delivered to its clients, the embedded Chrome browser has a user-agent with information on the corresponding phone model and Android version. On the other side, Firefox which can be downloaded from the Google Play Store does not contain this type of information because the store only offers a generic version for every Android mobile and it does not change its user-agent during its installation. Firefox indirectly provides a much better protection against fingerprint tracking by not disclosing device-related information.

You can find below two fingerprints collected from the same device but with a different browser: the first with Chrome, the second with Firefox.

```
Mozilla/5.0 (Linux; Android 4.4.4; D5803
    Build/23.0.1.A.5.77) AppleWebKit
    /537.36 (KHTML, like Gecko) Chrome
    /39.0.2171.93 Mobile Safari/537.36

Mozilla/5.0 (Android; Mobile; rv:34.0)
    Gecko/34.0 Firefox/34.0
```

V. Assessing the robustness of fingerprinting against possible technical evolutions

Web technologies evolve very fast, and we have seen in previous sections that some recent evolutions limit fingerprint-based identification (e.g., no Flash on mobile devices), while others open the door to increased identification (e.g., WebGL reveals fine grained information about the GPU).

In this section, we explore 6 potential evolutions that web technology providers (browsers and app developers, standardization organizations) could set up. We demonstrate that they would limit the effectiveness of browser fingerprinting by simulating their impact on our dataset. The first two scenarios are based on current trends in web technologies, while the others are more speculative and based on the observations made in previous sections. It should be noted that we do not estimate the impact of scenarios n°4 and 5 since we can hardly predict which attributes would be affected and how. We also treat scenario n°6 separately, due to its extreme nature.

Scenario n°1 - The definitive disappearance of Flash

The Flash plugin is progressively disappearing. It has been deprecated on all smartphones, tablets and mobile devices used to browse the web. On laptop and desktop browsers, Flash's security flaws have progressively created mistrust in its users. Click-to-play is becoming standard on most browsers. In the meantime, the number of web applications that replace Flash with JavaScript and HTML5 is also growing. These phenomena let us plausibly foresee the definitive disappearance of Flash.

Interestingly, Flash is still present in 80% of our Desktop fingerprints. Among these cases, 71.7% have it activated, 26.3% are using click-to-play protections, and 2.0% block Flash, likely by a browser extension.

Impact of scenario n°1: Figure 7 shows the impact of the Flash plugin on fingerprint uniqueness. The "No Flash" bar shows statistics over our complete dataset (for the 60,617 fingerprints that have Flash, we simulate its absence by removing the attributes obtained through Flash). The "Flash" bar is computed with the subset of fingerprints that have Flash, since it is not possible to simulate the presence of Flash on fingerprints that don't have it. We uniquely identify 95% of the browsers that have Flash, while this is reduced to 88% for those without Flash. The sizes of the anonymity sets are notably small, with less than 0.6% of the fingerprints in a set of size 50 or greater. These numbers confirm that browser fingerprinting in a Flash-less future is certainly possible, and that the wealth of fingerprintable attributes compensates for the lack of access to Flash specific attributes.

Scenario n°2 - The end of browser plugins

In 2013, Google decided to stop supporting NPAPI plugins in Chrome and to rely exclusively on the technology embedded

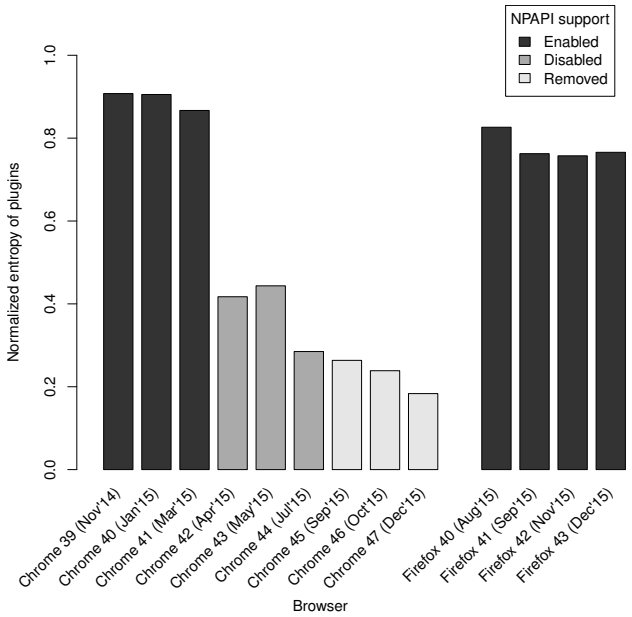


Fig. 8. Evolution of the normalized entropy of plugins for different browsers on desktop computers

in modern browsers and the functionalities offered by HTML5 and JavaScript to let developers extend the browser [14]. This has forced developers to migrate old plugins to newer alternatives [15] or to drop their support. Nevertheless, since its enforcement, it has the advantage of drastically reducing the entropy of the list of plugins. In 2015, version 42 of Chrome deprecated the support of NPAPI plugins by default and version 45 permanently removed their support.

This radical evolution, and the absence of plugins on mobile platforms, lets us foresee a more global evolution where browsers no longer provide a plugin-based architecture. Yet, this is challenging because plugins currently still provide a large number of features (as discussed in section II-B, we observed 2,458 different plugins in our dataset). Mozilla had plans to hide unpopular plugins with a whitelist [16] but they did not find a satisfying working solution that would not break websites or functionality. In October 2015, they announced the removal of NPAPI support by the end of 2016 [17].

Impact of scenario n°2: To estimate the impact of this scenario, we look at the entropy of plugins for Chrome since Google decided to deprecate the support of NPAPI plugins. Figure 8 shows the evolution of the normalized entropy of plugins for the stable releases of Chrome since the launch of the AmIUnique website. The last 4 stable versions of Firefox were added for comparison. Up to version 42, the normalized entropy of the list of plugins was above 0.8. Since the release of version 42, the entropy of the list of plugins has dropped below 0.5. This improvement is significant and the effects are getting bigger with the release of version 45 where the NPAPI support is permanently dropped (the entropy is not at zero since there are small differences in the plugin list between operating systems). Removing plugin support definitely impacts desktop fingerprints and it seems that their use in browser fingerprinting is becoming limited.

Scenario n°3 - Adherence to the standard HTTP headers

A major source of information for browser fingerprinting comes from application and system developers that add arbitrary information in headers by either modifying existing headers (e.g., the user-agent) or by adding new ones. Yet, the Internet Engineering Task Force (IETF) has standardized a list of fields for HTTP headers. The current diversity in the contents of the user-agent field results from a very long history of the 'browser wars', but could be standardized today. This scenario explores the possibility that technology providers converge on a standard set of HTTP header fields, and that they follow the standard.

Impact of scenario n°3: To estimate the impact of adherence to standard HTTP headers, we simulate the fact that they are all the same in our dataset. On desktops, the improvement is moderate with a decrease of exactly 8% from 90% to 82% in overall uniqueness. However, on mobile fingerprints, we can observe a drop of 21% from 81% to 60%. This illustrates the importance of headers, and especially the user-agent, for mobile fingerprinting and the fact that generic user-agents are essential for privacy.

Combining scenarios n°1-2-3: The biggest surprise of this analysis comes from combining the 3 scenarios. For mobile devices the results are significant but not overwhelming, the number of unique fingerprints drops by 22%. However for desktop devices, the percentage drops by a staggering 36%, from 90% to 54%. This means that if plugins disappear and if user-agents become generic, only one fingerprint out of two would be uniquely identifiable using our collected attributes, which is a very significant improvement to privacy over the current state of browser fingerprinting.

Scenario n°4 - Reduce the surface of HTML APIs

The potential disappearance of Flash and plugins will occur only if developers find suitable replacements with rich HTML and JavaScript features. Consequently, HTML APIs keep growing, providing access to an increasing number of information about the browser and its environment. As we saw in section III, the WebGL and canvas elements provide important information for identification. There are potentially many more APIs that leak identifying information.

Setting the best trade-off between rich features and privacy is a critical and difficult choice when setting up new APIs. Developers debate extensively on this kind of trade-off [18]. Yet, it is possible to foresee that future API developments, combined with informed studies about privacy such as the recent work by Olejnik and colleagues [19], will lead to reduced APIs that still provide rich features.

Scenario n°5 - Increase common default content

This scenario explores the possibility that browser or platform developers increase the amount of default elements,

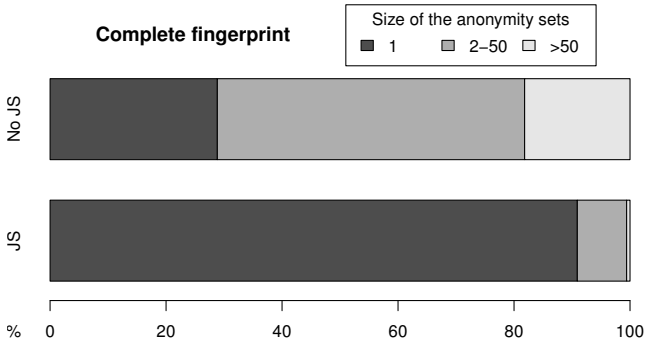


Fig. 9. Comparison of anonymity set sizes on the complete fingerprint between devices with and without JavaScript

which would be the only ones exposed publicly. For example, we could envision a whitelist of fonts that are authorized to be disclosed by the browser, as suggested by Fifield and Egelman [20]. Such a list would contain the default fonts provided by an operating system. This whitelist of fonts would also include a default encoding for emojis that is common to all versions of the operating system, or even common to all platforms.

This evolution would aim at reducing the amount of information disclosed to external servers. Yet, it should not prevent the users from adding new fonts or new emoji renderings. These customization decisions should be allowed without increasing the risks for privacy.

Scenario n°6 - The end of JavaScript

This last scenario explores the eventuality of coming back to a more static web, without JavaScript. This is the most unlikely today, as it would drastically reduce the dynamicity and comfort of browsing. Yet, there are currently millions of users who have installed the NoScript extension, which gives control to users on which websites JavaScript is allowed to run. We believe that it makes sense to explore the impact of such an evolution on identification through fingerprinting. Currently by disabling JavaScript, some sites do not render at all or render improperly, while most popular sites lose functionality even if properly rendered.

Figure 9 shows the impact of the unlikely return to a more static web. The presence of JavaScript in today's web helps make 89.4% of browsers uniquely identifiable, while removing JavaScript reduces the rate down to 29% on our dataset. This percentage could be even lower if user-agents become generic, as stated in scenario n°3. In that case, only 7% of fingerprints would be unique. The privacy benefits are undoubtedly significant but the cost to developers and to the users' comfort would be very high.

Conclusion

Here we have quantified the impact of possible technology evolution scenarii. While some of them could become reality in the not-so-distant future, others are less plausible. Yet, we demonstrate that they can benefit privacy with a limited impact on the beauty of current web browsing.

It is important to notice that tools already exist that can mitigate browser fingerprinting in similar ways as the scenarii discussed in this section. Ad and script blockers, like Ghostery [21] or Privacy Badger [22], prevent known fingerprinting scripts from being executed in the browser. The NoScript [23] extension blocks the execution of unwanted JavaScript scripts, which is a direct reflection of scenario n°6. The Tor browser team has modified Firefox to create a large range of defenses against browser fingerprinting [24]: from the complete removal of plugins to canvas image extraction blocking, their most recent addition being a defense against font enumeration by bundling a set of default fonts with the browser [25]. This protection illustrates scenario n°5 where the set of exposed fonts is greatly reduced.

VI. Related work

We distinguish three main areas of the literature on browser fingerprinting: analysis of client-side diversity, analysis of fingerprinting adoption on the web and server-side scripts, and advanced solutions to collect additional fingerprintable attributes. While our work is mostly related to the first category of work, we discuss the other two since they have inspired some of the fingerprinting techniques included in AmIUnique.org.

Client-side diversity: The work by Peter Eckersley is closely related to our study. In 2010 he launched the Panopticlick website, aimed at collecting device-specific information via a script that runs in the browser [7]. The script created browser fingerprints by collecting 10 different attributes that characterized the browser and its execution platform. He observed that 83% of visitors had instantaneously recognizable fingerprints, and this number rose to 94% for browsers that had the Flash or Java plugins enabled. He showed that the list of fonts (collected through the Flash API) and the list of plugins (collected through the JavaScript API) were the most distinguishable attributes.

The key novelties of our work with respect to Eckersley's study are as follow: the fingerprints we collect are richer and exploit some of the most recent web technologies (section III shows the essential role of canvas fingerprinting); Eckersley did not analyze mobile fingerprints separately from the others, while we perform a detailed analysis of how fingerprinting behaves for browsers on mobile devices; we assess the effectiveness of browser fingerprinting against different technological evolution scenarios. It should also be noted that the technological changes to the web since 2010 (e.g., the deprecation of the Netscape Plugin API, the steady disappearance of Flash, the arrival of HTML5) have strongly impacted browser fingerprinting, changing the importance of various fingerprintable attributes.

Very few other works have investigated the behavior of fingerprinting algorithms on client browsers. Yen et al. analyzed month-long datasets from Hotmail and Bing [26]. They combined the user-agent with the IP address, and succeeded in tracing back to a single host with 80% precision. While this work is also about fingerprinting, it has a much narrower

focus than ours (they consider only the user agent) and they do not consider the robustness of their approach, e.g., against agent spoofers. Spooren et al. recently analyzed 59 mobile device fingerprints [27] and concluded that "the fingerprints taken from mobile devices are far from unique". Our findings on mobile diversity are quite different (cf. section IV): 81% of our 13,105 mobile fingerprints are unique. We see two possibles reasons for the different conclusions: the scale effect (our dataset is two orders of magnitude larger that Spooren's); Spooren et al. do not consider canvas fingerprinting, while we demonstrate that the canvas test is essential to distinguish mobile fingerprints. Finally, Boda et al. [28] showed that cross-browser fingerprinting was feasible if enough data on the underlying operating system was collected. With our study, we did not explore this possibility since we do not know with certainty when two different fingerprints are from the same device but different browsers.

Adoption of fingerprinting on the web and server-side scripts: Some radically different works investigate the extent to which browser fingerprinting is adopted by web sites in the wild. Although these works investigate the same phenomenon as we do, the perspective is completely different, as are the conclusions and lessons learnt.

Nikiforakis et al. [1] analyzed the fingerprinting scripts of three popular commercial companies. They concluded that user-privacy was on "the losing side" and that commercial scripts used intrusive techniques to get the most data out of every browser.

FPDetective [2] was the first study about the adoption of browser fingerprinting on the web. Crawling the million most popular websites, they demonstrated the wide adoption of fingerprinting, and that fingerprinters completely disregard the user's Do Not Track preference. The same authors showed that 5.5% of the top 100,000 sites actively ran canvas fingerprinting scripts on their home pages [6].

New techniques for richer fingerprints: Several works have defined different ways to fingerprint devices or browsers in order to better differentiate them. Mowery and Schacham worked on the HTML canvas and WebGL elements [4], Mowery et al. on benchmarking the performance of core JavaScript operations [29], Mulazzani et al. checked the conformance of the browsers' JavaScript engines to the ECMAScript standard [30], Fifield et al. measured the onscreen dimensions of font glyphs [20], and Olejnik et al. used the HTML5 Battery Status API for fingerprinting purposes [19].

We kept only the work of Mowery and Schacha [4] in our script because canvas and WebGL tests are light and can be run in a matter of milliseconds. The other approaches take either too much time (e.g. more than 3 minutes to test the performance of JavaScript operations [29]), were too fragile (e.g., the battery API elements [19]), or did not add any valuable information to the pool of attributes that we already had (e.g. [20], [30]). We note that in general, new fingerprinting techniques are complementary to our work because they can be used as new distinguishing attributes in the fingerprinting algorithm, allowing for better precision in uniquely identifying browsers.

VII. Conclusion

In this work we analyzed 118,934 browser fingerprints collected through the AmIUnique.org web site. Our work focuses on the impact evolutions in modern web technology have had on the ability to uniquely identify devices through browser fingerprinting. We argue that modern web technologies provide a much improved user experience, albeit to the detriment of privacy.

The key insights from our study are as follows. First, our observations confirm the results of previous studies on the ease of fingerprinting in today's ecosystem [6], [31]. Second, we provide novel insights about the impact of the most recent browser APIs, including the first large-scale analysis of the HTML5 canvas on fingerprinting, as well as the influence of recent trends, such as the decreasing presence of Flash and other plugins on the web.

We also provide the first extensive analysis of fingerprints collected from mobile devices: 81% of the mobile fingerprints in our dataset are unique. We show that HTTP headers and HTML5 canvas fingerprinting play an essential role in identifying browsers on these devices. Furthermore, in the absence of the Flash plugin to provide the list of fonts, there is no longer any major discriminating attributes, thus identification is based on the collection of many lesser attributes that appear harmless by themselves, but when aggregated lead to unique fingerprints.

Our dataset, and the associated observations, allow us to evaluate the impact of possible evolutions in web technologies on browser fingerprinting. We show that certain scenarios would limit the detriment these technologies have on privacy, while preserving the current trend towards an ever more dynamic and rich web. Having generic HTTP headers and removing browser plugins could reduce fingerprint uniqueness in desktops by a strong 36%.

Acknowledgment

The authors would like to thank Nick Nikiforakis and Gildas Avoine for providing insightful feedback while writing this paper. We also want to thank our shepherd Adrienne Porter Felt and the anonymous reviewers for their valuable comments. This work is partially supported by the EU FP7-ICT-2011-9 No. 600654 DIVERSIFY and the CNRS INS2I JCJC 2016 FPDefendor projects.

References

[1] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless monster: Exploring the ecosystem of web-based device fingerprinting," in *Proc. of the Symp. on Security and Privacy*, 2013, pp. 541–555.

[2] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel, "Fpdetective: dusting the web for fingerprinters," in *Proc. of the Conf. on Computer & Communications Security (CCS)*. ACM, 2013, pp. 1129–1140.

[3] "Google Privacy Policy," http://www.google.com/policies/privacy/archive/20150501-20150605/.

[4] K. Mowery and H. Shacham, "Pixel perfect: Fingerprinting canvas in HTML5," in *Proceedings of W2SP 2012*, M. Fredrikson, Ed. IEEE Computer Society, May 2012.

[5] "Mobile internet usage soars by 67%," http://gs.statcounter.com/press/mobile-internet-usage-soars-by-67-perc.
[6] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, "The web never forgets: Persistent tracking mechanisms in the wild," in *Proceedings of the 21st ACM Conference on Computer and Communications Security (CCS 2014)*. ACM, 2014.
[7] P. Eckersley, "How unique is your web browser?" in *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, ser. PETS'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 1–18. [Online]. Available: http://dl.acm.org/citation.cfm?id=1881151.1881152
[8] "HTML Canvas 2D Context," http://www.w3.org/TR/2dcontext/.
[9] "Emoji and Dingbats," http://unicode.org/faq/emoji_dingbats.html.
[10] "WebGL Specification," https://www.khronos.org/registry/webgl/specs/latest/1.0/.
[11] "ANGLE: Almost Native Graphics Layer Engine," https://chromium.googlesource.com/angle/angle.
[12] "Masking Agent extension for Firefox," https://addons.mozilla.org/firefox/addon/masking-agent/.
[13] "An Update on Flash Player and Android," https://blogs.adobe.com/flashplayer/2012/06/flash-player-and-android-update.html.
[14] J. Schuh, "Saying Goodbye to Our Old Friend NPAPI," September 2013, https://blog.chromium.org/2013/09/saying-goodbye-to-our-old-friend-npapi.html.
[15] "NPAPI deprecation: developer guide," https://www.chromium.org/developers/npapi-deprecation — The Netscape Plugin API (NPAPI) has been permanently removed from Google Chrome since version 45. The Pepper API (PPAPI) is one option but few plugins exist and it is not proposed in the developer guide as an alternative.
[16] "Disallow enumeration of navigator.plugins (Mozilla bug tracker)," https://bugzilla.mozilla.org/show_bug.cgi?id=757726.
[17] "NPAPI Plugins in Firefox," https://blog.mozilla.org/futurereleases/2015/10/08/npapi-plugins-in-firefox/.
[18] "Extensive discussion about reducing the HTML battery API," https://groups.google.com/forum/#!topic/mozilla.dev.webapi/6gLD78z6ASI.
[19] L. Olejnik, G. Acar, C. Castelluccia, and C. Diaz, "The leaking battery: A privacy analysis of the html5 battery status api," Cryptology ePrint Archive, Report 2015/616, 2015, http://eprint.iacr.org/.
[20] D. Fifield and S. Egelman, "Fingerprinting web users through font metrics," in *Proceedings of the 19th international conference on Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer-Verlag, 2015.
[21] "Ghostery browser extension," https://www.ghostery.com/our-solutions/ghostery-browser-extention/.
[22] "Privacy Badger browser extension," https://www.eff.org/privacybadger.
[23] "NoScript browser extension," https://noscript.net/.
[24] "Design of the Tor browser," https://www.torproject.org/projects/torbrowser/design/.
[25] "Release of Tor with a new defense against font enumeration," https://blog.torproject.org/blog/tor-browser-55-released.
[26] T.-F. Yen, Y. Xie, F. Yu, R. P. Yu, and M. Abadi, "Host fingerprinting and tracking on the web: Privacy and security implications." in *NDSS*, 2012.
[27] J. Spooren, D. Preuveneers, and W. Joosen, "Mobile device fingerprinting considered harmful for risk-based authentication," in *Proceedings of the Eighth European Workshop on System Security*, ser. EuroSec '15. New York, NY, USA: ACM, 2015, pp. 6:1–6:6. [Online]. Available: http://doi.acm.org/10.1145/2751323.2751329
[28] K. Boda, A. M. Földes, G. G. Gulyás, and S. Imre, "User tracking on the web via cross-browser fingerprinting," in *Information Security Technology for Applications*, ser. Lecture Notes in Computer Science, P. Laud, Ed. Springer Berlin Heidelberg, 2012, vol. 7161, pp. 31–46. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-29615-4_4
[29] K. Mowery, D. Bogenreif, S. Yilek, and H. Shacham, "Fingerprinting information in JavaScript implementations," in *Proceedings of W2SP 2011*, H. Wang, Ed. IEEE Computer Society, May 2011.
[30] M. Mulazzani, P. Reschl, M. Huber, M. Leithner, S. Schrittwieser, E. Weippl, and F. C. Wien, "Fast and reliable browser identification with javascript engine fingerprinting," in *Web 2.0 Workshop on Security and Privacy (W2SP)*, vol. 5, 2013.
[31] " Technical analysis of client identification mechanisms ," https://www.chromium.org/Home/chromium-security/client-identification-mechanisms.
[32] "three.js official website, a JavaScript library to create 3D animations using WebGL," http://threejs.org/.

Appendix A
Normalized Shannon's entropy for all AmIUnique's attributes

Attribute	All	Desktop	Mobile
User agent	0.580	0.550	0.741
List of plugins	0.656	0.718	0.081
List of fonts (Flash)	0.497	0.548	0.033
Screen resolution (JS)	0.290	0.263	0.366
Timezone	0.198	0.200	0.245
Cookies enabled	0.015	0.016	0.011
Accept	0.082	0.082	0.105
Content encoding	0.091	0.089	0.122
Content language	0.351	0.344	0.424
List of HTTP headers	0.249	0.247	0.312
Platform (JS)	0.137	0.110	0.162
Do Not Track	0.056	0.057	0.058
Use of local storage	0.024	0.023	0.036
Use of session storage	0.024	0.023	0.036
Canvas	0.491	0.475	0.512
Vendor WebGL	0.127	0.125	0.131
Renderer WebGL	0.202	0.205	0.165
AdBlock	0.059	0.060	0.029

Appendix B
Our attempt at a WebGL test

As reported by Mowery et al. [4], the WebGL API can be used to render 3D forms in the browser. With the help of the three.js JavaScript library [32], we aimed to have a test that renders three different forms:

- a sphere
- a cube
- a Torus knot

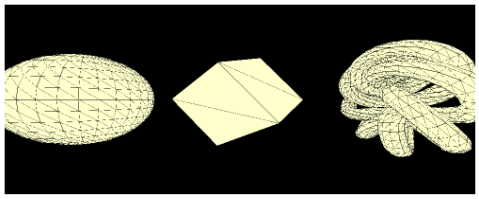
However, after analyzing more than 40,000 fingerprints, we concluded that the test was too brittle and unreliable to draw any conclusions from it. Indeed, if the user were to change the size of its browser window or open the browser console, the actual dimensions of the rendering context would be updated inside the library and the rendering would differ with just a simple page reload. Figure 10 shows three renderings of the same test with three different window sizes on the same device.

Appendix C
Additional Flash attributes

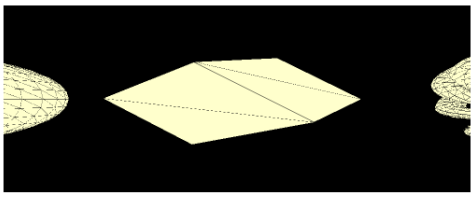
For Flash, we also collected the following four attributes:

- Capabilities.language
- Capabilities.os
- Capabilties.screenResolutionX
- Capabilties.screenResolutionY

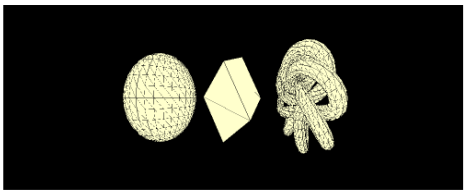
The language obtained through Flash is the devices main language, but it is not as precise as the content language header collected through HTTP. For the screen resolution, it can be more interesting than the JavaScript value because Flash will return the full resolution of a multi-screen setup and not the resolution of a single screen. Finally, when analyzing the data from the string collected from the OS property, it confirmed what has been observed by Nikiforakis et al. [1] in 2013. Depending on the OS and the browser, the information is often generic, returning "Windows" or "Linux", but in some cases



(a) 1920x1200 window



(b) 960x1200 window



(c) 1080x600 window

Fig. 10. Different renderings of the WebGL test on the same device

it returns the type of the OS with the exact version of the kernel (for example, “Mac OS 10.8.2" or “Linux 3.18.4-1-ARCH"). This level of detail could be used to forge an attack against a vulnerable system, and it is surprising that little has changed since it was originally reported. In the end, we did not keep this information for our study because it did not increase the number of unique fingerprints and would mainly serve to detect inconsistencies (e.g., caused by User-Agent spoofers).

TABLE IV
STATISTICS OF ADDITIONAL FLASH ATTRIBUTES

Flash attribute	Distinct values	Unique values
Screen resolution XxY	584	329
Language	44	10
Platform	968	483

APPENDIX D
COMPARISON TO THE PANOPTICLICK STUDY

To complement section 2.3.2 of our paper that compares our dataset with the one from Panopticlick [7], we recreated the same graphs to show the impact of 5 years of browser development on browser fingerprinting.

A. Distribution of fingerprints

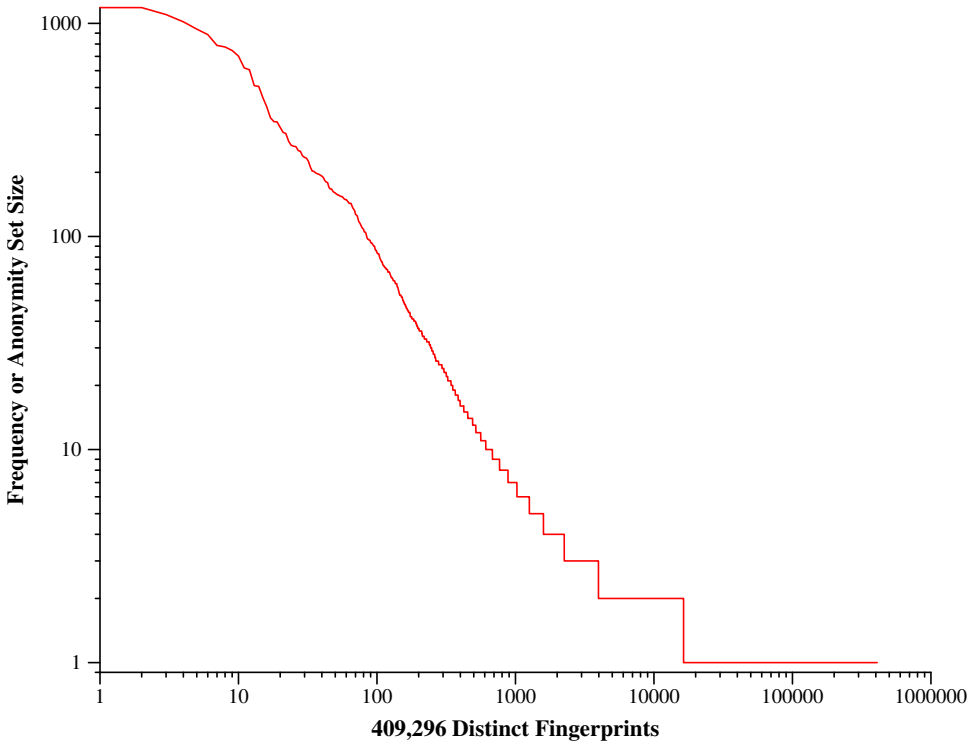
If we compare both frequency distributions in Figure 11 w.r.t. anonymity set sizes, we can observe that the overall trend is similar in both graphs with set sizes quickly dropping to 1. While Panopticlick has 83.6% of its fingerprints located on the tail on the right of Graph 11a, AmIUnique presents a slightly lower number on Graph 11b with 79.4% of fingerprints that are unique in the database (fingerprints with and without JavaScript).

B. Distribution of browsers

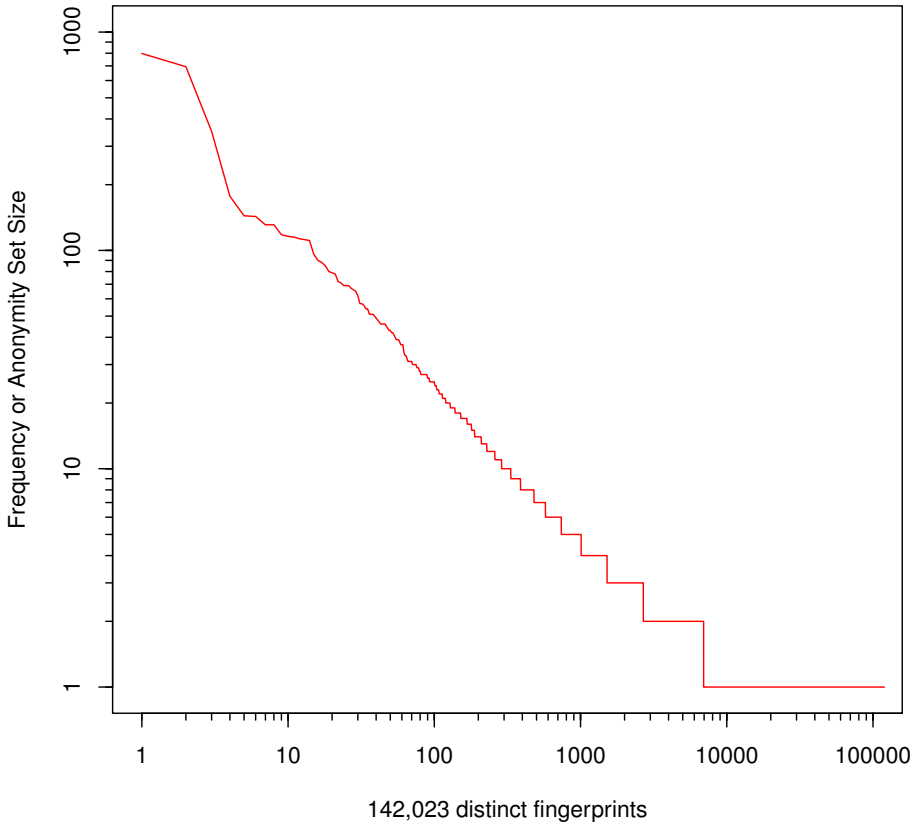
Figure 12 shows the distribution of surprisal for different categories of browsers. We can see that the overall trend is similar in both graphs. The main noticeable difference is the number of browsers in each category. While the Panopticlick dataset was constituted of mainly Firefox browsers followed by Chrome and Internet Explorer, our dataset put Chrome and Firefox at the same level with all the other browsers behind. This shows the rapid growth of the Chrome userbase over the last 5 years and the decline of Internet Explorer.

C. Anonymity set sizes

Figure 13 shows the size of anonymity sets for all attributes if we consider them independently from each other. In our case, the bigger an anonymity set is, the better it is for privacy. If a value is in an anonymity set of size 1, it means that the observed value is unique and is not shared by another fingerprint. With all the attributes that we collected on AmIUnique, we could not add all of them in Figure 13b for readability reasons so we focused on attributes with the highest level of entropy. If we look at the upper left part of both Figure 13a and Figure 13b, we observe very similar results and the most discriminating attributes on AmIUnique are still the same as the ones observed by Eckersley (mainly fonts and plugins) but with the addition of new efficient techniques like canvas fingerprinting (see section III-A of the paper for more information).

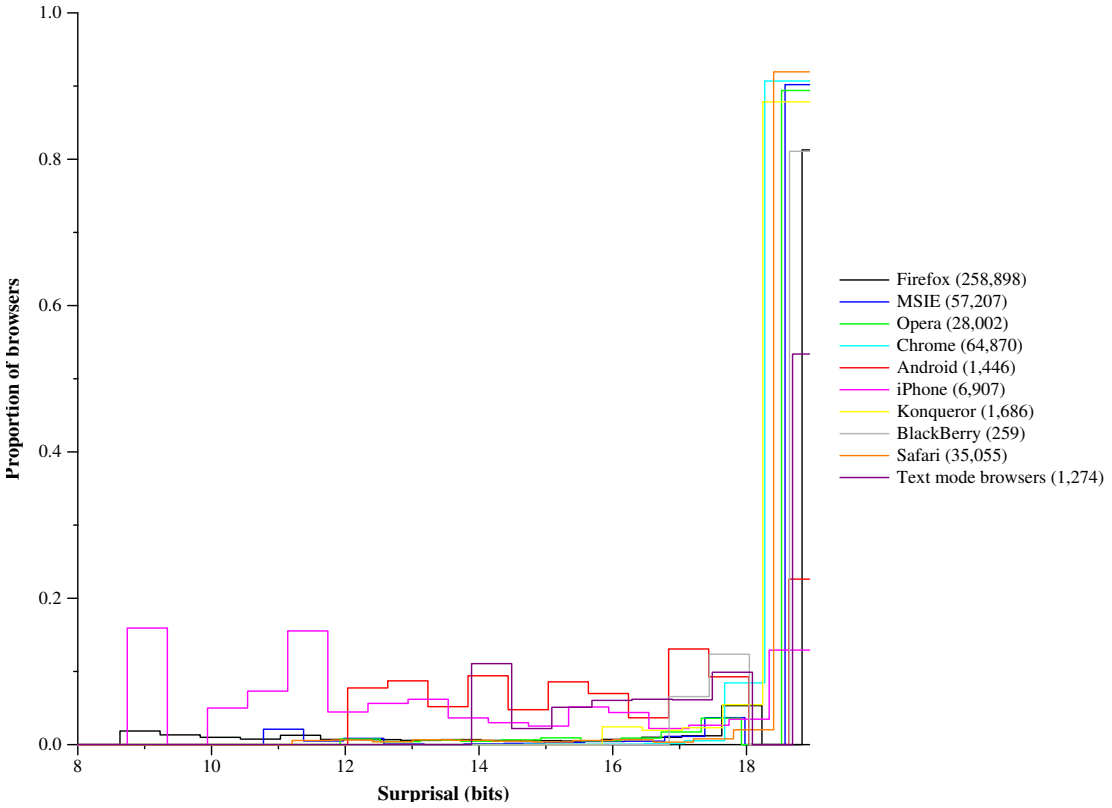


(a) Panopticlick distribution (Fig. 1 of [7])

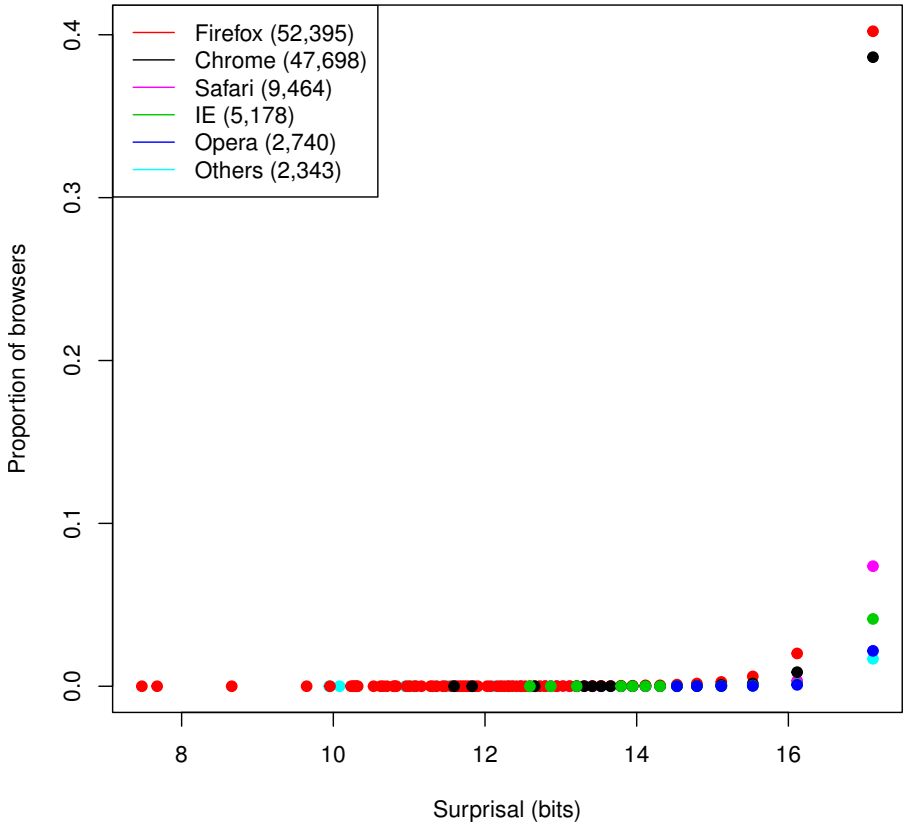


(b) AmIUnique distribution

Fig. 11. Distribution of fingerprints w.r.t. anonymity set size

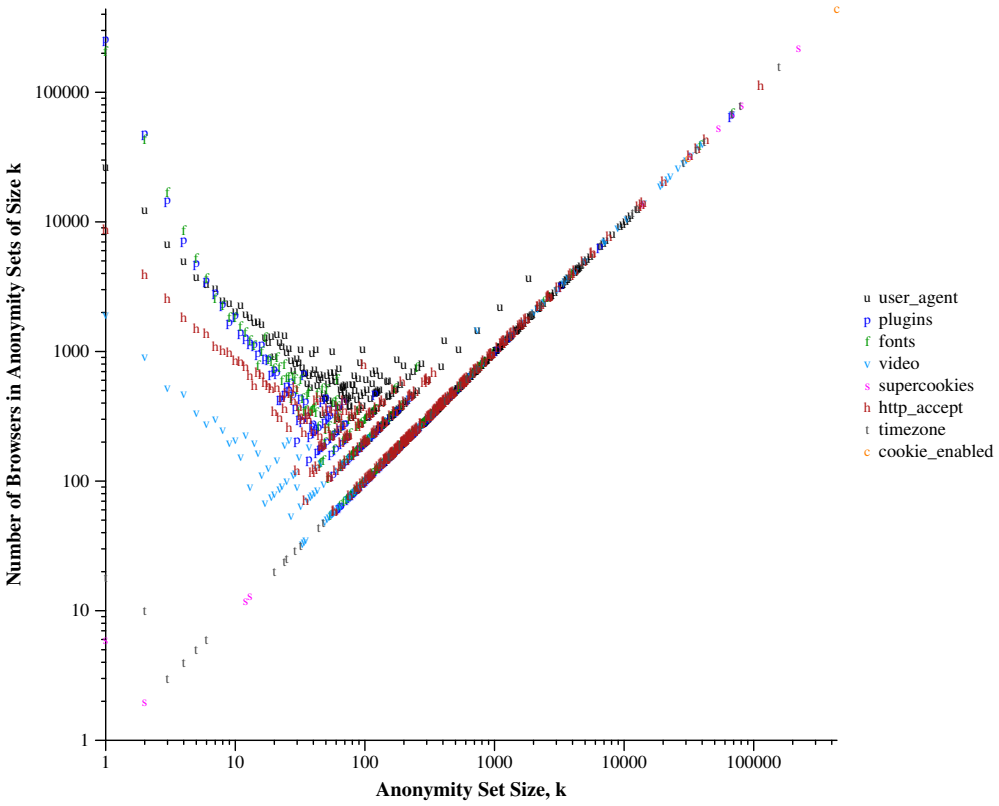


(a) Panopticlick distribution (Fig. 2 of [7])

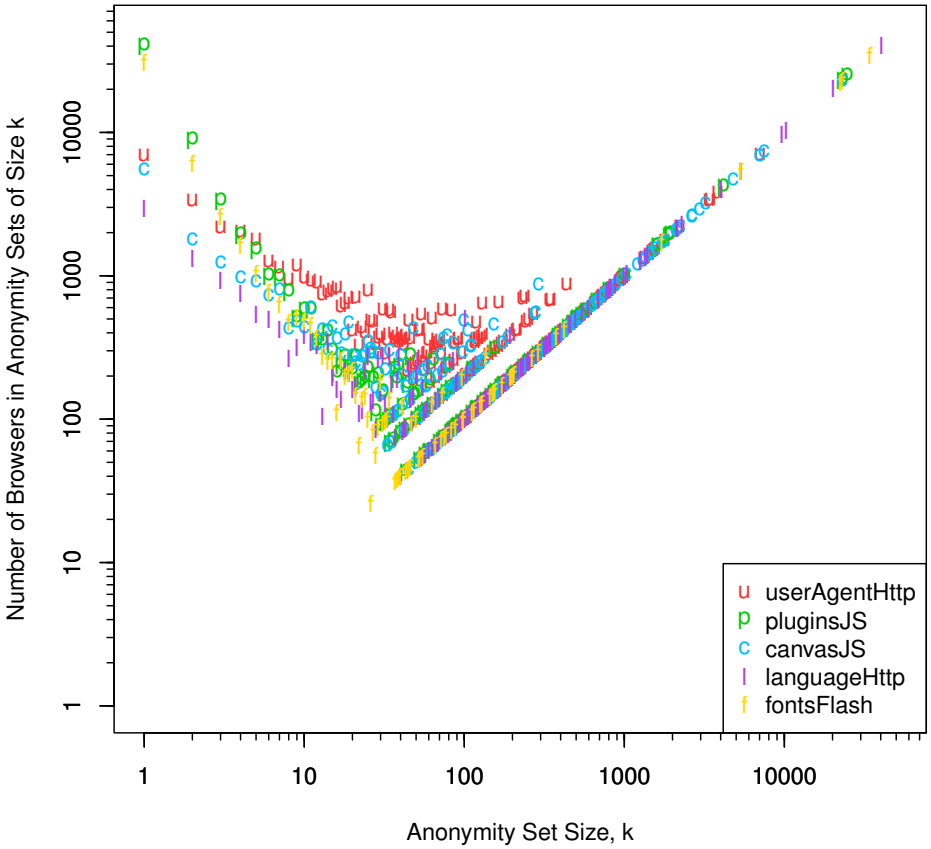


(b) AmIUnique distribution

Fig. 12. Surprisal distributions for different categories of browser



(a) Panopticlick distribution (Fig. 3 of [7])



(b) AmIUnique distribution

Fig. 13. Number of users in anonymity sets of different sizes, considering each variable separately