Browser Fingerprinting and the Importance of Digital Privacy

Aaron M. Corpstein
corps002@morris.umn.edu
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

Abstract

Browser fingerprinting is a type of internet tracking where the attributes of a user's computer and browser accessing a web page are remotely recorded and then used for profiling, tracking, and advertising purposes. This paper focuses on defining browser fingerprinting and enumerating ways in which the user can combat fingerprinting. Browser fingerprinting can be thwarted by changing attributes within the user's browser or machine, using a browser designed to combat fingerprinting, or with security and anti-fingerprinting focused browser extensions. All of these methods are capable of increasing the security of the user.

Keywords: Browser Fingerprinting, Cookies, Browser, Digital Privacy, Internet Tracking

1 Introduction

In modern times the internet has become an increasingly important element of daily life, with the expansion of the internet, the privacy of users is in increasing danger [11]. Browser fingerprinting is a relatively new method of tracking users across the internet and across separate sessions, endangering user privacy, via recording and storing data about the user remotely. The type of data collected tends to be about the browser the user is accessing the internet on and its specifications, the specifications of the machine accessing internet, and various other attributes about the machine/user. The data captured this way is not saved anywhere where the user has access to and is subject to the will of the website or company that recorded the data. This data is used for tracking a user's internet activity in the interest of product or political advertising. The study done by Gómez-Boix et al [7] offers a new way of combating browser fingerprinting in a way that is conscious of the user experience. Another study done by Alan et al [4] goes into depth on the effectiveness of browser fingerprinting and the effect of client diversity on fingerprinting.

2 Background

To understand the impact of browser fingerprinting on user privacy it is necessary to define the browser, APIs, cookies, and with a general definition of browser fingerprinting.

2.1 History of the Internet Browser

The modern Web Browser was born out of the need for accessing the growing online network. In 1991 the networking protocol HTTP (Hypertext Transfer Protocol) was developed to standardize communication over the network so that the method of connection is not tied to the type of machine seeking access to the network [1, 10] setting the groundwork for modern internet protocols.

The communication system that enables users to access the internet is the Client-Server model. The model includes three nodes: the server, the client and a database. The client is an interface the user uses to access a service offered by a server. This interface is usually a browser which is also referred to as a 'web client'. The server is the location where an internet service is provided, the server can also be referred to as the 'web server'. The server is where hosting of a 'web page' takes place and is where computation can occur. The server may also communicate with a database. The database is where data is stored for access by the server which can also be passed to the user through the server.

The client-server model accomplishes its goal of providing a service to the user through the use of the 'request-response' model. The communication in the request-response model takes the form of the client requesting information from the server and the server responding with the correct data and instructions for loading the page.

With the implementation of the HTTP protocol, a header was included in all request-response transmissions. The header is referred to as the "User-Agent header", it provides information that allows any client to "talk" to any server over the HTTP protocol. Whenever a user accesses a web page, the User-Agent header is communicated in the requests sent to the web server [3]. The User-Agent Request header contains a token that indicates compatibility with Mozilla [1], the operating system of the computer, the Gecko Version [2], and the browser version respectively. An example of a browser fingerprint can be observed in Figure 1.

[1] the token is used mainly for historical reasons; Netscape became Mozilla and they developed one of the first 'browsers'. Mozilla then became open source and the platform from which most browsers are based on

[2] Gecko is a web driver that renders web pages, another invention of the Mozilla foundation

```
Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:85.0)
     Gecko/20100101 Firefox/85.0
```

Figure 1. User-Agent Header of a browser on a Linux platform.

2.2 APIs

API stands for Application Programming Interface. An API is what handles interaction between software applications or between hardware and software. This concept extends to internet applications in the form of Web APIs. Web APIs handle interaction between web servers and web clients. The APIs common to fingerprinting include; the Canvas API, the WebGL API, and the Web Audio API.

The Canvas API is an API that houses methods that help in drawing and manipulating graphics on the web client. Such manipulation is done within a canvas drawing area where users can draw, animate shapes, and render text. The WebGL API is specifically designed to render user interactive three-dimensional objects on the web client without the need for additional JavaScript plugins. The Web Audio API is designed to handle audio operations. Browser fingerprints can be created from these APIs by collecting data on the existence of the APIs on a client and data on the rendering/antialiasing done on the object rendered via the APIs [10]. Any API can also be queried in the JavaScript return a set of strings that are base64 representations of images rendered using the image APIs. These strings are long and unique to each client.

2.3 Cookies

Cookies are bits of data stored locally on a web client to increase efficiency and improve the user experience. Common uses for cookies include storage of a user id for login or form information that would be filled into form fields: addresses, card numbers, names, and the like. However, another application for cookies was found; cookies could be used for tracking purposes by a web server or by third parties like advertisers. Tracking cookies take the form of cookies that record the browsing history of the user. With cookies being useful in such a manner, they have been used in browser fingerprinting for creating highly accurate fingerprints but are not necessary for successful fingerprinting [6]. The accuracy of fingerprints can be improved by simply storing the fingerprint on the client for reference in later sessions [9], where "session" describes the instances in which a browser is booted up or logged into then used for a period of time before being shut down or logged off.

2.4 Browser Fingerprint Definition

In 2009, the computer scientist Jonathan Mayer explored the differences inherent to individual web clients for deanonymization. What he found was that the differences in browsers could be observed and recorded by a server to uniquely identify users. In an experiment he ran he found that by recording a list of the *Navigator plugins*, he could uniquely identify web clients with about a 96.3% accuracy. Navigator plugins are plugins that are returned by the 'Navigator interface'. The Navigator interface represents the general state of the client, while Navigator itself is a handler for various parts of the client i.e. plugins, language, cookies, and the hardware. Navigator can be queried by scripts running on the web client to return a list of plugins that are on the web client[2]. Mayer's experiment used about 1.3 thousand clients with 1278 of them being uniquely identifiable. Later in 2010 another experiment was conducted by Peter Eckersley and it found that fingerprints could be created via the User-Agent Header and the JavaScript plugins like Flash or Java. This brought about the term "Browser Fingerprint" and a first set of attributes that make up a browser fingerprint.

Browser fingerprints are usually formed on the client. This is done through the use of scripts running on the web client while the user in accessing a web page that makes use of fingerprinting. Many of these scripts are in the scripting language: JavaScript. These scripts run on the web client and record various attributes about the client which then can get sent to a server and stored in a database.

Not only can the host of the web server run tracking scripts on the web client, but third parties can also run their own scripts alongside anything that might be running on the web page (this however can be mitigated through changing certain settings within most browsers). Third party fingerprinting creates the possibility for cross-site fingerprinting where advertising companies can run scripts on multiple sites, identifying individuals on each site.

3 Methods of fingerprinting

Here the methodology used for browser fingerprinting and the impact of cookies is discussed.

3.1 Methods Used for Browser Fingerprinting

Most, if not all, methods for browser fingerprinting employ collection of data about the user's client to form a digital fingerprint that can be used to distinguish one browser from another and by extension one user from another. This data includes information about the machine the browser is running on, the specifics of the peripherals the browser has access to, the configuration of the browser itself, and any cookies that are present on the browser. All these digital markers will be discussed in this section.

3.1.1 Types of data Collected for a Fingerprint. The data collected by fingerprinting is the "data regarding the configuration of a user's browser and system when this user visits a website" [7]. This data is referred to as the attributes of the browser. The User-Agent header was one of the first attributes used in creating fingerprints and is always present

in any transmission to a web server. Figure 2 is an example of a fingerprint that was collected using a tool that allows users to record a real fingerprint of their browser [5].

Figure 2. Browser Fingerprint of an Ubuntu Machine.

The tool used to create the fingerprint in Figure 2 captures a plethora of details about the user's browser and the machine the browser is running on. Some of the more notable attributes captured in the fingerprint is the Operating System, the screen resolution, language and time zone, and usage of certain APIs. The rest is a list of fonts used by the browser, the hardware concurrency (number of cores in the processor), the color depth (number of bits available for pixels), the 'audio' is a hash value of how the computer handles audio output, if and how the machine handles touch screen capabilities, how many 'touch points' (the start and end points of the touch capable area on the screen), and the session and local storage records whether or not the browser has access to storing memory during the session and storing data that can be accessed over multiple sessions. All of the attributes captured can then be stored and used to track users across the internet. The fingerprint can be stored on a server then sold to an advertisement company and the fingerprint can be stored on the user's machine in the form of a cookie. The fingerprint will remain stable and associated with the browser that accessed the web page as long as the browser or machine configuration does not change.

3.1.2 How Cookies are Used in Fingerprinting. As mentioned, cookies can be utilized in browser fingerprinting. The fingerprint that is stored in a cookie is stored on the user's browser for future identification. The cookie version is used in conjunction with a fingerprint that is stored on the host's server to more effectively track users that visit web pages that employ fingerprint tracking. Cookies are useful in making the identification process more accurate, but they are not necessary. A cookie is not necessary because, a fingerprint can still be recorded from the client and then that fingerprint is checked against other fingerprints that the server has stored to identify users. Simply, a cookie offers stability in the process.

4 Countering Fingerprinting

There are two main methodologies when combating browser fingerprinting, both stem from either changing the attributes of the web client or obscuring the attributes from the web page's server. All methods vary in effectiveness of thwarting fingerprinting and impact to the user's experience.

4.1 Changing Attributes

Browser fingerprints can utilize a great many attributes concerning the configuration of a user's browser. Most fingerprints look like the one captured in Figure 2, with the most common attributes being the User-Agent header, language, color depth (how many bits the pixels' values have access to), screen resolution, time zone, platform, Java plugins and JavaScript, Canvas API, and enabled cookies. Of the attributes listed, some of them are changeable by the user via the browser's settings and/or the settings of the operating system. The attribute the user has the most agency in changing are the system or browser language, the color depth, time zone, if Java or JavaScript is enabled in the browser, whether or not cookies are enabled, and the screen resolution. So, as it turns out a significant portion of the set of attributes in a common fingerprint can be changed. This, however, is not as promising as it would seem due to the usability factor after changes are made. For instance, many web pages require the use of JavaScript to appear and run correctly on the client. The same can be said for disabling cookies: many sites require cookies to be enabled so the user can remain logged in. Also changes made to attributes like the screen resolution and language can negatively affect the user's ability to interface with the browser. The user experience post direct attribute manipulation is then the reason for why the researchers involved with the study conducted [7] attempted to create a browser that did not impact the overall user experience.

4.2 Browser Based Measures: Extensions

Browser extensions have and are currently being used to defend the user against various vectors of internet tracking. The extensions mentioned here combat browser fingerprinting in multiple fashions, all with varying success. These browser extensions can be found on many of the popular browsers and can help in protecting user privacy.

4.2.1 Script Blocking. Script blocking is a method used to hinder tracking scripts and browser fingerprinting. Script

blocking is a method used among the following popular browser extensions: Ad-Block Plus[3], Disconnect[4], Ghostery[5], NoScript[6], Privacy Badger[7], and uBlock[8]. All these extensions disallow certain scripts originating from the web pages to be ran on the browser accessing said web pages. More specifically, Ghostery and Privacy Badger block scripts via a blacklist that gets updated periodically. Additionally, Privacy Badger can also use analysis of heuristics to block third-party scripts hidden within web pages. NoScript can block scripts similar to Ghostery and Privacy Badger but can also allow user specified scripts to be ran in the browser, giving the user even more control over what is running in the browser. Script blocking is a good security measure. However, script blocking can be seen by the party that placed the scripts in the web page and this can even be included as an attribute to a fingerprint [6].

4.2.2 Attribute Blocking. Attribute blocking is a method employed by some extensions to narrow the number of attributes given off by a web client to shroud the client from the web server and third parties. Attribute blocking is accomplished by simply blocking the web server's access to certain attributes. In doing so, the fingerprint emitted by the web client is less unique and therefore less traceable and less identifiable. Browser extensions that employ this style of counter are CanvasBlocker[9] and Canvas Defender[10]. Both extensions work by blocking access to the Canvas API and to HTML5. However, there is a flaw in this approach, it is detectable. The FP-Scanner test suite can detect when the Canvas Element is being altered and therefore can detect usage of these types of extensions that block certain attributes since these extensions block the Canvas API [7]. This is important because if this countermeasure is detected, the behavior can then be recorded and included as part of the fingerprint.

4.2.3 Attribute Switching. Attribute switching employs randomization of two sets of attributes to 'lie' to the web server. What gets randomized is the list of plugins that gets seen by the web server to prevent scanning by JavaScript. The other attribute that is altered is the User-Agent Header. Attribute switching is accomplished by User-Agent Spoofing extensions, User Agent Switcher[11] is such an extension. The values that can be change are a token that indicates compatibility with Mozilla, the Platform (the Operating System) the web client is running on, the Gecko Version, and the browser version. In addition to switching values in the User-Agent header, the User Agent Switcher extension can also automatically switch User-Agents based on URLs at the discretion of the user. The attributes changed in this way are changed randomly. The Random Agent Spoofer extension offers the user the functionality to make Browser configuration Profiles that can be swapped to automatically based on user specified URLs similar to the User Agent Switcher extension. The Random Agent Spoofer extension's browser configuration profiles include all common attributes as described in 3.1.1. The extension also blocks other fingerprinting methods that leverage the Canvas, WebGL, and other similar APIs. Overall Attribute Switching can be a quite powerful method to protect against browser fingerprinting [7].

4.2.4 Attribute Blurring. Attribute blurring is an offshoot of Attribute Switching. Attribute Blurring like its counterpart aims to change the attribute values to produce a fingerprint differing from the actual fingerprint the web client would emit. Attribute blurring accomplishes its goal by introducing noise (changing parts at random) in attributes where noise can be injected. An attribute where noise can be added are the Canvas and Audio elements. In both the Audio and Canvas elements there is an id associated to the element, within the id, noise can be added to change the emitted fingerprint. The browser extension FPGuard[12] utilizes Attribute Blurring to create random or preset fingerprint emitted by the web client [7].

4.3 Non-Unique Fingerprints

In the study done by Gómez-Boix et al [7], the researchers devised a clever solution to fingerprinting. The solution would employ aspects of attribute switching but not at the cost of the user experience. The approach taken by Gómez-Boix et al [7] was to create a browser fingerprint that does not obscure or just randomly change the fingerprint given off by the web client but rather to assign each browser a fingerprint that is non-unique to be seen by fingerprinting. These non-unique fingerprints would be shared by multiple users in a pool so that the privacy of the individual is maintained.

The study that creates the concept of non-unique fingerprints was introduced by an earlier study by a similar group. In the earlier study done by Gómez-Boix et al [8], research was conducted on how many unique fingerprints can be discovered in larger sets of fingerprints to strengthen non-unique fingerprints so that the fingerprints do not become unique if attributes are changed or added. What was found was that in a large sample of fingerprints (about 2 million) that were collected from 15 French websites, only about 33.6% of the fingerprints were able to be uniquely identifiable, as opposed to previous studies where the accuracy was much higher with smaller sample sizes [8].

[3] https://adblockplus.org/
[4] https://disconnect.me/
[5] https://www.ghostery.com/
[6] https://noscript.net/
[7] https://privacybadger.org/
[8] https://ublock.org/
[9] https://github.com/kkapsner/CanvasBlocker
[10] https://multilogin.com/canvas-defender/
[11] http://useragentswitcher.org/

[12] FPGuard is not available for download. FPGuard was created as a solution by FaizKhademi, et al.

In the experiment conducted by Gómez-Boix et al [7], the researchers used clustering algorithms that group together sets of similar fingerprints to produce final sets of fingerprints. The fingerprints used in the clustering process were acquired from a data set that was produced in the earlier study by Gómez-Boix et al [8] and would be partitioned into partitions based on the operating system and browser. This resulted in four partitions; Linux/Firefox, Linux/Chrome, MAC OS X/Firefox, and MAC OS X/Chrome. The Clustering algorithms utilized were K-Means, Density Based Clustering and Agglomerative Hierarchical algorithms. All of which are common but outside the scope of this paper. The team also did a second round of clustering where aggregation was applied to produce more clusters. From the clusters generated, the centroid [13] and the fingerprint with the lowest entropy [14] value were extracted as the non-unique fingerprint that in practice would be applied to a user's browser.

The distance function used to produce the clusters measures the similarity between fingerprints. In order to compare fingerprint, the attributes were given weight values based on importance where the sum of the weights per fingerprint equal one. F stands for fingerprints, d stands for distance, weight values denoted as w_i, and a stands for the attributes 3.1.1 of the fingerprints.

$$D(F_1, F_2) = \sum_{i=1}^{n} w_i * d(F_{1a_i}, F_{2a_i})$$

To measure the effectiveness of the non-unique fingerprints, two measures were devised that would then be used for a series of graphs to judge the fingerprints on. The first of the two is the *Identifiability* metric I. K stands for the number of partitions or clusters. $u(c_k)$ is the number of devices within the cluster k [7]. I is then bounded on the interval $[K^2/U, U]$ where U is the total number of devices in the cluster. I then reaches its minimum value when all the clusters ideally reach a maximum number of devices defined by: $u(c) = U/K$. Identifiability is a score given to a cluster of fingerprints as it is derived from the function.

$$I = \sum_{k=1}^{K} \frac{1}{u(c_k)}$$

The third function finds the *Disruption* within the non-unique fingerprints. The disruption is the distance between the non-unique fingerprint extracted from the cluster and a fingerprint extracted from a target/user's client. The function takes into account the number of attributes altered between the two fingerprints and then the number of fingerprints required to generate the non-unique fingerprint where $u(FP_i)$ is the number of fingerprints that share the fingerprint i. Better clusters have lower disruption.

[13]The centroid being the center of the cluster

[14]Entropy determines the how well the cluster forms

$$R = \sum_{k=1}^{K} \sum_{i \in c_k} (d(FP_i, FP^{\{k\}}) * \frac{u(FP_i)}{U})$$

4.4 Results of Non-Unique Fingerprints

The research conducted by Gómez-Boix et al found that in fact the approach did reduce the identifiability of users when the non-unique fingerprints were applied [7]. The team first took a measure of the identifiability of the raw data in the data set. This can be observed in the figure 3

Dataset	# of FPs	Number of devices	% of unique	Ident.
Linux, Chrome	1,176	4,117	67.6	921.05
Linux, Firefox	804	2,316	61.3	594.31
Mac OS X 10.12, Chrome	1,047	1,769	73.6	871.24
Mac OS X 10.12, Firefox	3,202	3,832	88.6	2,991.81

Figure 3. Identifiability of the raw data

In the figure it can be seen that the number of uniquely identifiable fingerprints is quite high, reaching 88.6% for Mac OS X 10.12 with Firefox. This resulted in an identifiability score of 2,999.81 which is far higher than the scores after clustering 4,5.

In the first experiment, the researchers produced the results that can be seen in figure 4. The disruption is shown on the y-axis, the identifiability is shown on the x-axis. The numbers along with each point represent the number of clusters k, the number of fingerprints in the smallest cluster, and the number of devices within the smallest cluster respectively. The first experiment produced promising but not sufficient results so a second experiment was conducted.

The results of the second experiment can be seen in Figure 5. This time aggregation was added as another layer of processing to the clustering scenarios. This produced better results with the identifiability score being less than 1 now and the disruption remaining quite low.

These findings can be supported by the research done in a study by Alan et al where the effects of client diversity in a population was examined [4]. What was found was that the accuracy of a fingerprint relies on certain attributes more than others, specifically when the browser or operating system is changed. In the study, the fingerprints used were pre-generated fingerprints, meaning that they were not from real machines but randomly created. In the study the researchers had five test scenarios where the accuracy of the fingerprints were tested. The scenarios are:

1. The client went unchanged
2. Same browser and operating system, different device
3. Same browser, different operating system
4. Different browser, same operating system
5. Both the browser and operating system changed

Of the scenarios tested, the two that had diminished the accuracy of the fingerprints the most were when just the

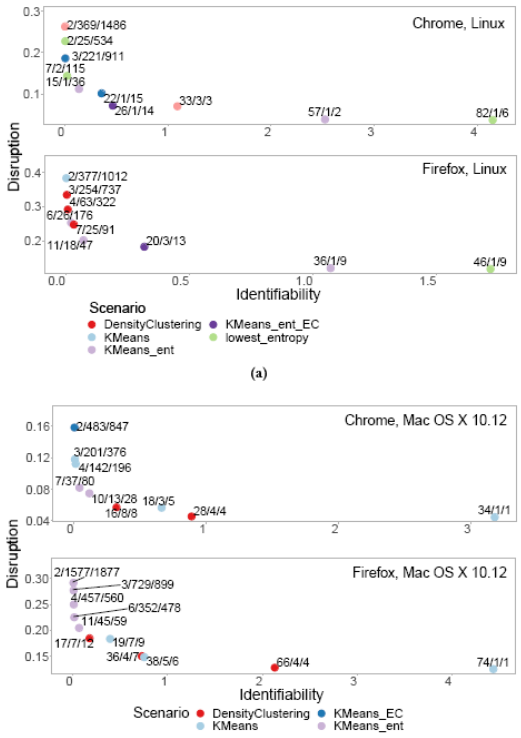


Figure 4. Results from the first experiment [7]

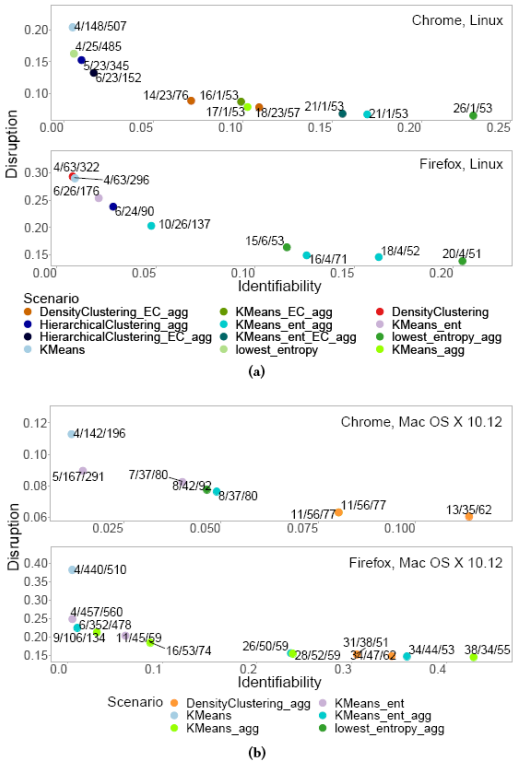
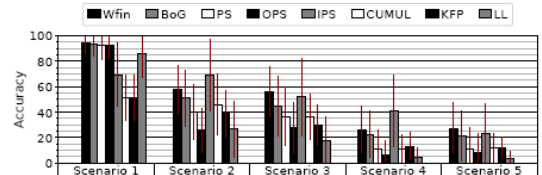


Figure 5. Second set of results [7]

browser changed and when both the browser and OS was changed. The scenario that has a more realistic application is the one where only the browser changed between sessions. These results can be seen in figure 6, where each bar on the graph is a fingerprinting technique. This is important because this shows that meaningful action can be taken at the browser level to combat fingerprinting.

To make use of the findings by Gómez-Boix et al [7], the non-unique fingerprint could be applied with the use of what the researchers dubbed a "fingerprint protection platform". This platform would collect a fingerprint from the user's client and then return a recommended fingerprint from the set of fingerprints generated by the clustering analysis. The platform could take the form of a new browser, a website or even Docker[15]. With the findings in both studies, it would appear that the usage non-unique fingerprints can be a really practical solution to protect user's privacy while also preserving the user experience.



	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5
Wfin	94	57	55	26	27
BoG	93	51	44	22	21
PS	92	40	36	11	11
OPS	92	25	27	6	8
IPS	69	68	52	41	23
CUMUL	51	46	35	10	11
KFP	51	39	29	13	11
LL	85	26	17	4	3

Figure 6. Results of Client Diversity [8], where Wfin, BoG, ect. are fingerprinting methods

5 Conclusions

Browser fingerprinting is a technique of de-anonymizing and tracking internet users that has a large impact on the privacy and security of the internet. What has been discussed was the attributes that are used in fingerprinting a user's web client for de-anonymization, some current solutions offered by various browser extensions, and an experimental solution that intends on providing security at little cost to the user experience. The usage of non-unique fingerprinting could be the next major step in protecting the user against browser fingerprinting. As stated, a "fingerprint protection platform" would need to be developed to make the security offered by non-unique fingerprints accessible. In the future, steps to protect user privacy should be taken and the solutions found could be used to ensure the privacy of individuals on the internet.

[15]Docker is a software container program where code can be ran in isolated OS-level virtual environments.

Acknowledgments

I would like to thank my family for providing support, Elena Machkasova for advising me through the entire writing process, and Adam Casey for his early advice.

References

[1] Mozilla Foundation . *Evolution of HTTP.* Mozilla Foundation. Accessed: 2021-04-12.

[2] Mozilla Foundation . *Navigator interface.* Mozilla Foundation. Accessed: 2021-03-31.

[3] Wikipedia . *User Agent.* Wikipedia. Accessed: 2021-03-28.

[4] Hasan Faik Alan and Jasleen Kaur. 2019. Client Diversity Factor in HTTPS Webpage Fingerprinting. In *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy* (Richardson, Texas, USA) *(CODASPY '19)*. Association for Computing Machinery, New York, NY, USA, 279–290. https://doi.org/10.1145/3292006.3300045

[5] Rob Braxman. . *Browser Fingerprint Generation Tool.* Brax.me. Accessed: 2021-04-1.

[6] Rob Braxman. . *What Browser to Use? About Browser Isolation.* Youtube. Accessed: 2021-04-1.

[7] Alejandro Gómez-Boix, Davide Frey, Yérom-David Bromberg, and Benoit Baudry. 2019. A Collaborative Strategy for Mitigating Tracking through Browser Fingerprinting. In *Proceedings of the 6th ACM Workshop on Moving Target Defense* (London, United Kingdom) *(MTD'19)*. Association for Computing Machinery, New York, NY, USA, 67–78. https://doi.org/10.1145/3338468.3356828

[8] Alejandro Gómez-Boix, Pierre Laperdrix, and Benoit Baudry. 2018. *Hiding in the Crowd: An Analysis of the Effectiveness of Browser Fingerprinting at Large Scale.* International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 309–318. https://doi-org.ezproxy.morris.umn.edu/10.1145/3178876.3186097

[9] Andrew J. Kaizer and Minaxi Gupta. 2016. Towards Automatic Identification of JavaScript-Oriented Machine-Based Tracking. In *Proceedings of the 2016 ACM on International Workshop on Security And Privacy Analytics* (New Orleans, Louisiana, USA) *(IWSPA '16)*. Association for Computing Machinery, New York, NY, USA, 33–40. https://doi.org/10.1145/2875475.2875479

[10] Pierre Laperdrix, Nataliia Bielova, Benoit Baudry, and Gildas Avoine. 2020. Browser Fingerprinting: A Survey. *ACM Trans. Web* 14, 2, Article 8 (April 2020), 33 pages. https://doi.org/10.1145/3386040

[11] Wade L. Robison. 2018. Digital Privacy: Leibniz 2.0. *SIGCAS Comput. Soc.* 47, 4 (July 2018), 134–144. https://doi.org/10.1145/3243141.3243155