Fingerprinting for Web Applications: from Devices to Related Groups

Christine Blakemore[1], João Redol[2], Miguel Correia[1]

[1]*INESC-ID, Instituto Superior Técnico, Universidade de Lisboa* [2]*EyeSee, Lda*
christine.blakemore@tecnico.ulisboa.pt, joao.redol@eyesee.pt, miguel.p.correia@tecnico.ulisboa.pt

***Abstract*—Identifying users and user devices is as important in web applications as in many other contexts. In web applications, user identification usually involves an authentication process, e.g., providing a username and a password. Identification is also possible without explicit authentication using cookies or device fingerprints. Device fingerprinting is also useful for other purposes, e.g., to serve as a second factor of authentication. Recently some interest appeared in the problem of cross-device fingerprinting, i.e., of the identification of the same user in different devices using fingerprinting. We target a variation of the problem that we call *related group fingerprinting*. We define a related group as a set of persons (e.g., a family) that share the same home network. We devised a related group fingerprinting scheme that we evaluated experimentally with data from hundreds of users. This evaluation suggests that group fingerprinting is feasible.**

1. Introduction

Identifying users and user devices is as important in web applications as in many other contexts. In web applications, *user identification* usually involves an *authentication* process, e.g., providing credentials like a username, a password, or a token-generated code. Identification is also possible without explicit authentication using cookies or device fingerprinting.

Device fingerprinting, or web-based device fingerprinting, consists in gathering multiple pieces of information from the client's device and browser for identification purposes, e.g., the fonts and plugins installed, the model, version, and language of the browser, etc. [1], [2].[1] Device fingerprinting is useful for several purposes. It can serve as a second or third factor of authentication [4], complementing the above-mentioned credentials. It can also serve to detect lost or stolen user devices, when they contact a server that uses this form of authentication. Both examples are extremely important today, with the increasing number of phishing attacks and thefts of personal devices.

We use the term *fingerprint* to designate a set of features that identifies a device, browser, or user. To be useful, fingerprints have to combine information that allows to uniquely identify devices. The more diverse the feature's values, the more they may be unique, as their values may be less likely shared by multiple devices. Every time a user accesses a web page that includes fingerprinting software, the device fingerprint is collected and compared to a database of known devices. If the device is not in the database, it is added, increasing the number of known devices [5], [6].

Recently some interest appeared in the problem of *cross-device fingerprinting*, i.e., of the identification of the same user accessing a website or web application from different devices (e.g., tablet, laptop, smartphone) without explicit authentication. However, this problem is challenging and may require collecting personal user data (e.g., mouse dynamics data), so we consider a variation of the problem that we call *related group fingerprinting*. We define a *related group* as a set of persons (e.g., a family) that share the same home network. Related group fingerprinting consists in detecting if a device belongs to a related group. We aim to solve this problem without using personal data.

In terms of applications, related group fingerprinting is useful for a spectrum of use cases. Usually there is some level of trust inside related groups, so related group fingerprinting may be used to assign levels of risk to certain operations without explicitly knowing who the user is. For example, it may be used as an additional form of authentication in certain online operations, like accessing homebanking services or school websites.

We did an experimental evaluation of this scheme with data from hundreds of users. First we developed two websites for volunteers to access. The websites contained both a short questionnaire and mechanisms to extract fingerprints. Then we studied mainly three aspects:

(1) Is it still possible to perform *device fingerprinting* given the very recent privacy mechanisms in web browsers that prevent the extraction of browser history data, one of the most powerful fingerprinting mechanisms [7]? We concluded that it is still possible and studied the minimum subset of features required (Section 4.3).

(2) Given the fingerprints collected, is it possible to perform *cross-device fingerprinting*, i.e., to recognize if it is the same user in different devices? The answer was negative (Section 4.4).

(3) Is it possible to perform *related group fingerprinting*? The answer was, this time, positive (Section 4.5).

1. There is a related problem also called device fingerprinting that aims at doing active identification of computers by sending them packets, e.g., [3]. Our problem is different: to identify devices that access a web server.

The *related group fingerprinting* scheme was designed based on the analysis of the experimental data obtained (see Section 3). For privacy reasons, this data does not include personal data like biometric data or browser history. Similarly to other fingerprinting schemes, each time a user accesses the website, a fingerprint is extracted and stored, in case the device fingerprint is not yet in the database. Moreover, the *group* of the device is also obtained and stored in case it had not been stored before, i.e., it is not in the set of groups of that particular device. A group is essentially a network to which a device is connected to. The group identifier is: the IP of the device of the used browser as observed by the server, in case the IP is owned by an Internet Service Provider (ISP); or the name of the organization obtained from the *whois* command, in case it is a company, governmental agency, etc. Notice that a device can belong to several groups, for example, if it is a smartphone connected to a home network, to a 4G network, and to an office network, which are 3 different groups.

The contributions of this paper are the following: (1) the definition of the problem of related group fingerprinting; (2) a scheme to perform related group fingerprinting; (3) an experimental study of this scheme; (4) an experimental study of device and cross-device fingerprinting.

2. Context and Related Work

This section briefly surveys existing work on web-based fingerprinting. *Browser fingerprinting* is commonly used as part of anti-fraud and advertisement systems, as it may avoid the need of authentication with credentials provided by the user and the use of cookies [5]. Similarly to cookies, fingerprinting can be used both for legitimate and illegitimate ends. Legitimately, it may be used to combat fraud, confirming if someone who is trying to login into a web-based server is in fact a legitimate user rather than an attacker who got hold of stolen credentials. It can also be used to combat click fraud, i.e., illegitimate clicking of advertisements to increase payment [8]. Illegitimately, it may be used to track the actions of a person, violating his privacy in some sense. The EFF (Electronic Frontier Foundation) has implemented a fingerprinting algorithm, Panopticlick, which anonymously logs the configuration and version information of user operating system (OS), browser, and plugins, and compares it to a database of many other users' configurations [9]. The goal is to understand how identifiable users are, and evaluate the capabilities of Internet tracking and advertising companies who try to record user's activities. In this experience, 94.2% of the browsers using Flash or Java were considered unique in the sample used.

Much user data, such as fonts installed or cookies, can be obtained using JavaScript, which is an object-oriented scripting language used for various purposes as interactive web contents, asynchronous communication and dynamic document content alteration [10]. Allowing dynamic content to be executed on web browsers grants developers the ability of creating rich and interactive web interfaces. By enabling asynchronous communication with servers, it is possible to constantly update data without a page refresh. JavaScript provides APIs that grant access to device and browser-related properties. JavaScript has become a powerful fingerprinting tool [5]. It allows obtaining information such as architecture, OS language, system time, and screen resolution. The two JavaScript APIs that have been mainly investigated and exploited for fingerprinting purposes are: (1) *Navigator object* that represents the properties of the device and browser environment (browser name and version, supported plugins and MIME types, and OS and browser architecture); *Screen object* that contains information about the settings of the device's screen displaying the browser (screen resolution, and color and pixel depth). JavaScript has been used for fingerprinting in other ways, exploiting performance [6], the HTML5 canvas [11], and browsing history [7], [8].

The previous techniques may be designated as browser fingerprinting, in the sense that they aim to identify a browser running on a device. The problem of *device fingerprinting* or *cross-browser fingerprinting* is different because there may be several browsers installed on a device. There is some preliminary work on doing such cross-browser fingerprinting that aims to identify a device irrespectively of the browser being used [12]. These techniques rely on features that have to be independent of the browser, for example: OS name and version, layout engine type and version, list of fonts, screen resolution, and time zone. That work shows that it is possible to do cross-browser (or device) fingerprinting based on such features. It also shows that their scheme is resilient to the modification of one of the features, i.e., that the scheme still identifies devices correctly if one of the individual fingerprints changes. To the best of our knowledge there is no work on cross-device or related group fingerprinting.

There has been some work on preventing device tracking and fingerprinting due to privacy concerns [13]. TrackingFree is an anti-tracking browser system that, instead of blocking the use of identifiers like cookies and flash files, isolates these identifiers into different units or browser principals, blocking third-party tracking [14]. PriVaricator uses a policy randomization process [15]. For offset measurements as offsetHeight, offsetWidth and getBoundingClientRect, instead of returning the original offset value, the proposed policy approach is to return, respectively, zero, a random number between 0 and 100, and the original offset value with $\pm 5\%$ noise. This approach both generates plausible offset values as well as creates enough noise to confuse fingerprinting. Bloom cookies use Bloom filters as a privacy-preserving data structure to provide a better tradeoff between privacy, personalization, and network efficiency [16].

3. The Fingerprinting Approach

This section presents our approach. Section 3.1 presents our device fingerprinting scheme, as it is the basis of our related group fingerprinting scheme. Section 3.2 discusses how the problem of cross-device fingerprinting might be solved. Finally, Section 3.3 presents our related group fingerprinting scheme.

3.1. Device Fingerprinting

With the help of JavaScript we are able to extract multiple properties from user devices. As previously mentioned in Section 2, the cross-browser fingerprinting method of [12] used features such as lists of installed fonts, OS version, and screen resolution. That work concluded that it was possible to create a unique fingerprint based on a set of these features and that, even if one of them changed, the scheme would still identify the device. This suggests that some of the fingerprints may change with time and, in fact, this is true for most of them. For instance, the OS version tends to change due to system updates.

3.2. Cross-Device Fingerprinting

Cross-device fingerprinting is about identifying the same user using different devices. If cross-browser fingerprinting requires features that are independent of the browser, cross-device fingerprinting requires features that are common across devices (e.g., tablet, laptop). This is equivalent to say that the fingerprints have to be related with who the user is and how he acts. We can envisage three options:

(1) Fingerprints with configurations that the users use across devices: our experience suggested that this would not work, still we evaluated experimentally if it was possible to create fingerprints with such data (Section 4.4).

(2) Fingerprints with static biometric data (fingerprints from fingers, face geometry, iris, etc. [17]): such biometric data is used for authentication so it would work, but we are interested in schemes that do not require the user to authenticate and that are not blunt violations of privacy, so we excluded them.

(3) Fingerprints with dynamic biometric data (keystroke dynamics [18], mouse dynamics [19], etc.): this would be possible but it would require that the user interacted with the application for some time for collecting the fingerprint, making it application-dependent, not very practical, and problematic from the privacy angle, so we discarded it.

3.3. Related Group Fingerprinting

As already explained, the purpose of related group fingerprinting is to identify if a device belongs to a *related group*, i.e., to a group of persons who live together, e.g., a family. In practice, we define a related group as a set of persons that share the same home network. We use the term *group* to designate a set of devices that share the same company or organization network. A device may belong to several groups and related groups (e.g., a company network, and a home network).

We assume that home networks use private IP addressing, i.e., are behind a network address translation (NAT) router. Therefore, the identifier of a related group is the public IP address of the NAT router, which is the sender IP observed by the application's web server in the IP datagrams it receives from the device. Using, for instance, the *whois* command it is possible to understand if an IP address belongs to an ISP, so if the corresponding group is a candidate for being considered a related group. The identifier of a group that is not a home group is the name of the network, e.g., the content of the field *netname* returned by the *whois* command. The name cannot be an IP address, as the company or organization may have many public IP addresses.

The decision if a group is a *related group* can be based on one or more criteria, for instance:

(1) *ISP-provided IP address:* a related group must have an IP address assigned by an ISP as observed in *whois*, as we do not envisage a related group having another kind of IP address;

(2) *Limited number of devices:* a related group cannot contain more than D_{thresh} devices (e.g., 15), to exclude other networks with an IP owned by an ISP (e.g., the ISP network itself);

(3) *Used mostly out of business hours:* accesses from the network are mostly made out of business hours (at night, during weekends), again to exclude other networks.

Algorithm 1: Related group fingerprinting algorithm for collecting a fingerprint (executed by the server)

```
Input: HTTP request $M$ received from the client
if $M$ is the initial request from the client then
    add fingerprinting scripts to the reply;
else
    if $M$ contains fingerprint then
        $F \leftarrow get\_fingerprint(M)$;
        if $F \notin database$ then
            $G \leftarrow get\_group(M)$;
            $add\_device(database, F, G)$;
        else
            $add\_group\_to\_device(database, F, G)$;
```

The related group fingerprinting scheme is represented in Algorithm 1. The algorithm uses Criterion 1 and is executed by the server when a message with a fingerprint is received from a browser. The algorithm does not return anything, only adds data to the server database. The main types of queries to that database are the following:

- Is group G a related group?
- To which related group(s) does the device with fingerprint F belong to?

There are a few pathological cases that deserve some discussion:

(1) What happens if the IP address of a home network with users accessing the web application changes? In that case, the server will keep data about two related groups, identified by the old and the new IP addresses, although the related group is actually the same. We do not envisage this to be problematic to most applications, if these events are not too frequent, which is the case in major ISPs.

(2) What happens if two different devices from unrelated persons have the same fingerprint? In that case the database will recognize the two as a single device that will belong to a set of groups that contains the groups of both devices. We do not expect this to be problematic if these collisions are rare.

4. Experimental Evaluation

This section presents the experiments we did in order to understand the problem and to assess our related group fingerprinting scheme. We start by presenting the testing tool and the datasets we obtained.

4.1. Testing Tool and Datasets

In order to test and analyse our fingerprinting mechanisms, we needed to find a way of collecting an experimental dataset from a group of users willing to contribute. We came up with the possibility of creating a website with which users could interact. They would access this website as a very simple and quick task, allowing us to collect all the necessary features for the fingerprint. In fact, two different websites were developed. The first aimed to interact with as many users as possible, so it collected both fingerprints and requested the users to answer a few questions. This experiment generated *Dataset I*. We later registered, for this same dataset, a total of 410 distinct users who willingly participated in the experiment. The second website had a different purpose, which was to study the fingerprints stability and did not require users to answer any questions. Only 5 users participated in this experiment, that consisted in accessing that website for two weeks using the installed browsers on their devices. The data collected from this experiment resulted in our *Dataset II*. Both websites provided the browsers JavaScript code to collect information from the devices.

Table 1 shows the fingerprinting features obtained using the two experimental websites and a brief description (Fp1 is not shown as it designates the timestamp, that is not really a feature). Dataset I is composed of 531 fingerprints, whereas Dataset II has a total of 168 fingerprints collected from the above-mentioned 5 users. This means that we had a total of 699 complete accesses to the two websites. The data was collected in January and February 2016.

In order to perform the study, we needed to know which fingerprints belonged to the same user, therefore the websites asked for data that was then used to create a user ID. The purpose was to have data that uniquely identified the user without his privacy being compromised. Therefore, we asked for two pieces of reasonably harmless personal data – the users first name and the last 3 digits of his cellphone number – and calculated a SHA-256 hash of that data in the browser [20]. This hash results in the user ID and is sent to the server (unlike the name and digits that do not leave the browser at all), guaranteeing the confidentiality of the user data. This ID is used to compare and confirm results, identifying which results belong to the same user.

TABLE 1. FINGERPRINTING FEATURES COLLECTED IN THE EXPERIMENTS

Ref.	Feature	Description
Fp2	Plugins	Plugins installed in the browser, e.g., Java and Flash
Fp3	UserAgent	HTTP header field that identifies the browser and provides system details
Fp4	Browser	Browser used when accessing the website
Fp5	Cookies Enabled	True or false according to the cookies being enabled or disabled in the browser
Fp6	Display	Color depth, pixel depth, screen width and height
Fp7	System Fonts	List of installed system fonts
Fp8	Browser Lang	Language preferred by the browser
Fp9	OS	Operating system in which the browser is running
Fp10	Time Zone	Time zone of the browser based on its geo-location
Fp11	Touch	True or false whether the device has touch screen
Fp12	IP Address	IP of the device (public address if NATed)
Fp13	Latitude	Latitude of the browser (if user allows browser to provide the location)
Fp14	Longitude	Longitude of the browser (same condition)
Fp15	HTTP Accept	HTTP header field that gives the accepted media types (for the HTTP response)
Fp16	HTTP Accept Encode	HTTP header field that gives the accepted content encodings
Fp17	HTTP Accept Language	HTTP header field that gives the set of preferred natural languages
Fp18	Platform	Platform of the browser, e.g., MacIntel or Win32
Fp19	Do Not Track	Do Not Track enabled or disabled in the browser

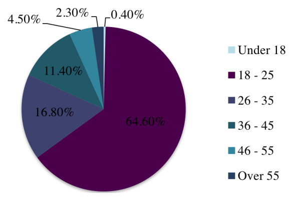
Out of the 531 entries of Dataset I, 410 different IDs were registered. In other words, 410 users participated in this experiment. Also, from a total of 531 data entries, 506 devices were classified as being unique, meaning that only 25 entries (4.71%) were repeated for a certain device (indicating a different browser). To study and analyze the retrieved user data, a testing tool was developed. This tool calculates parameters such as the Hamming distance between two data entries and the entropy values for each feature of the fingerprint.

Users who accessed the website had to answer a few questions for statistical purposes. Users were questioned about both personal information (e.g., age group, gender) and information concerning device/browser interaction. Figure 1a shows that over 60% of the participants were between 18 and 25 years old. We can also see that almost 3/4 of the participants were male, in Figure 1b. Figures 1c, 1d, 1e and 1f show the percentage of answers for the rest of the questions.

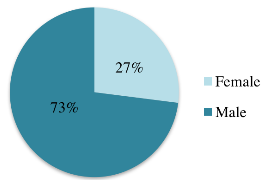
4.2. Metrics

This section presents four metrics we use to analyze the experimental data: Hamming distance, entropy, precision, and accuracy.

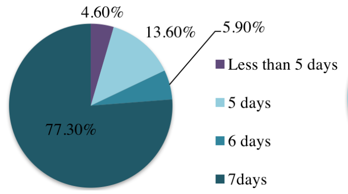
We use a generalization of the *Hamming distance* to measure the difference between two fingerprints, or the error in case they are two fingerprints of the same device [21]. The Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. We consider a fingerprint to be a vector of feature values, 18 values in our case (Table 1). For use with fingerprints, we generalized the metric so that what is compared are not symbols, but the feature values. For the collected 531 data entries, we created a 531x531 matrix



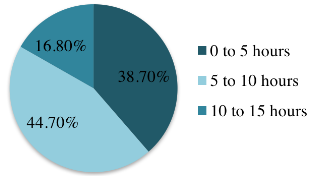
(a) Participants age group



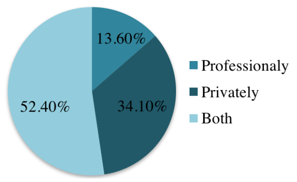
(b) Participants gender



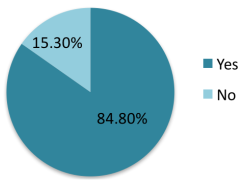
(c) Q1: How many days per week do you use this device? (%)



(d) Q2: Number of hours per day that you use this device? (%)



(e) Q3: How do you use this device? (%)



(f) Q4: Is this the browser you use the most (on this device)? (%)

Figure 1. Statistics for Dataset I

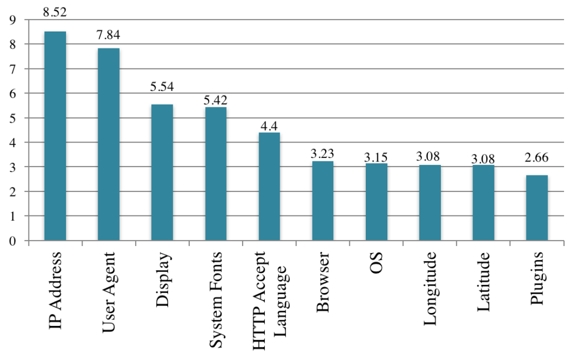


Figure 2. Features with top 10 entropy values

TABLE 2. ENTROPY VALUES

Fingerprint Ref.	Entropy Value (bits)	Num. Unique Values
Fp2	2.66	28
Fp3	7.84	330
Fp4	3.23	29
Fp5	0.04	2
Fp6	5.54	143
Fp7	5.42	129
Fp8	2.33	17
Fp9	3.15	17
Fp10	0.41	8
Fp11	0.96	2
Fp12	8.52	415
Fp13	3.08	155
Fp14	3.08	154
Fp15	1.18	4
Fp16	0.54	6
Fp17	4.40	80
Fp18	2.32	9
Fp19	0.88	6

where each position contains the Hamming distance value between two different fingerprints.

The *entropy* allows us to calculate how unique a certain feature is based on the amount of information it contains [9], [12], [22]. Using this metric it is possible to understand which combination of features produces unique fingerprints and those that are more useful to reach that uniqueness [22]. To calculate the entropy for each fingerprint feature, we use Shannon's entropy formula [23]. For a given feature f, the probability of ocurrence of the i^{th} value ρ_i, and the number of fingerprints n (with $n = 531$ for Dataset I), the entropy H or the produced information is given by (in bits): $H(f) = -\sum_{i=1}^{n} \rho_i \times \log_2 \rho_i$. For example, if there was a feature f' with different values for all 531 browser fingerprints of Dataset I, it would have the maximum entropy of $H(f') = 8.97$ bits. Furthermore, a fingerprint for that dataset might be composed of that single feature, as it would uniquely identify every browser and device.

To evaluate the importance of features in the fingerprint, we calculate their entropy for Dataset I. The higher the entropy the more unique the fingerprint is. Figure 2 shows the 10 features with the highest entropy values. The maximum entropy value corresponds to the IP address and is 8.52 bits, meaning that if we randomly choose a browser from the 531 browsers, it will share the same IP address with at most one browser in the other $2^{8.52} = 367$ browsers. As we can see in Figure 2, after the IP address, the user agent, display properties and fonts have the highest entropy values. However, entropy is not the single consideration to take into account when selecting features to compose fingerprint; e.g., the IP address is a dynamic feature so it is not adequate.

The question we are interested in is essentially: *given two fingerprints F and F', does F' correspond to the same entity (device, browser, or user, depending on the case) than F?* We say that there is a *false positive* (FP) if the fingerprinting mechanism says the entity is the same (a positive P), that there is a match (Hamming distance is zero), and this is false. We say that there is a *false negative* (FN) if the mechanism says the entities are different (a negative N) and it is the same. In the case of *related groups* the relevant condition is not being the same entity but to belong to the related group, otherwise the idea is the same. Given these definitions we are interested in two metrics [24]. The first is the *precision*, which measures the confidence we can have when the mechanism says it is the same entity, there is a match. The second is the *accuracy*, which measures in some sense the correctness of the mechanism in terms of the rate between correct matches and the total (we abuse the notation and use TP to mean the number of true positives, P for the number of positives, etc.): $precision = \frac{TP}{TP+FP}$ and $accuracy = \frac{TP+TN}{P+N}$.

4.3. Device Fingerprinting

For device fingerprinting, we aimed to answer the following questions:

Question 1 – *Device Fingerprinting:* Is it possible to uniquely identify a certain device using only the features of Table 1?

This question is related to the challenge we mentioned of current privacy mechanisms in web browsers preventing the use of some of the most efficient features, namely browsing history [7]. Given these restrictions we limited the fingerprinting features to those of Table 1.

To determine whether it is possible or not to perform device fingerprinting, we have developed a program that calculates both the Hamming distance and the entropy values for each fingerprint feature. This same program also determines how many unique devices exist in the chosen dataset (Dataset I), and returns a list with these devices.

The program identified 506 unique devices, as there were 25 pairs of entries for which the Hamming distance was zero. Therefore, there were no false positives or false negatives. This means that the precision and the accuracy were both 1.

Question 2 – *Fingerprinting Resilience:* Does the fingerprint of a certain device tolerate changes in features?

This question is related to the issue of some of the features changing with time. The objective is to understand if these changes still allow fingerprinting.

To test the fingerprinting resilience we used the same program from Question 1. The difference was in the used data sample, as we created 3 new datasets based on Dataset I. Each of these datasets is identical to Dataset I except that we removed one of the following features at a time: System Fonts, OS, and Plugins. For the 3 datasets there were no false negatives (FN), as the mechanism never failed to identify two fingerprints corresponding to the same device as being so. However, there were false positives (FP) as there was some confusion in identifying some of the devices.

The best results were found with the exclusion of the OS feature, with 506 unique devices, which corresponds to a precision and accuracy of 1. By excluding plugins, we also obtained a very good result, with 505 unique devices and 1 false positive, which means a precision of 0.998 and an accuracy of 1. Lastly, without system fonts, 491 unique devices were detected, 15 false positives, which gives a precision of 0.97 and an accuracy of 1.

This allow us to conclude that with Dataset I the device fingerprinting mechanism tolerates well the change of one feature, as even the lowest precision is quite high (0.97). With changes to more than one feature results would be necessarily worse.

Question 3 – *Fingerprint Stability:* Are there any significant changes in the data over the days? Are the features more or less static?

With the purpose of studying the stability of the fingerprint, we used Dataset II, and tried to determine if it presented any changes over time. For the 16 devices present in this dataset, the only features that changed were the IP address and the geo-location (latitude and longitude), as some of the devices accessed the website from different networks and places. This is good as stability is needed for device fingerprinting and related group fingerprinting to work as expected.

TABLE 3. FINGERPRINT FEATURE SUBSETS

Subset	Features in the Fingerprint	Unique Devices	Precision
1	Fp3, Fp6, Fp7	358	0.72
2	Fp3, Fp6, Fp7, Fp17	457	0.91
3	Fp3, Fp6, Fp7, Fp17, Fp4	487	0.96
4	Fp3, Fp6, Fp7, Fp17, Fp4, Fp9	487	0.96
5	same as above plus Fp2	505	1.00

Question 4 – *Minimum Subset:* What is the minimum subset of fingerprint features able to successfully identify a user in Dataset I?

Another interesting matter is to understand it we can find a suitable subset of features able to identify a user. For this we used Dataset I and the features with the highest entropy levels (Table 2), with the exception of the IP address, longitude and latitude (which are the most dynamic, making them bad candidates). Table 3 registers the number of unique devices and the precision for five different subsets of features, for a total of 531 data entries. Clearly the minimum subset depends on what is considered acceptable precision.

4.4. Cross-Device Fingerprinting

In relation to cross-device fingerprinting we have a single question:

Question 5 – *Cross-Device Fingerprinting:* Using Dataset I, is it possible to perform cross-device fingerprint, i.e., to identify the same user behind two devices using only the features collected?

With the features we were able to collect, we cannot tell if two distinct devices do in fact belong to the same user. If two devices have different operating systems, for instance, the idea of these devices belonging to the same user is already rejected. This is the usual case as users have more than one device but they are different, e.g., a laptop running Windows and a tablet running Android. Therefore, the collected features are indeed insufficient to successfully perform a cross-device fingerprinting identification.

4.5. Related Group Fingerprinting

This section evaluates the related group fingerprinting scheme presented in Section 3.3. For this purpose we enhanced both datasets with data about the IP owner obtained using the *whois* command. In this study we considered that a group is a related group using a single criterion of those we listed: the IP address being provided by an ISP or not (Criterion 1). The extended Dataset I emulates the database of the related group fingerprinting scheme.

Question 6 – *Related groups:* What groups and related groups exist in Dataset I and what are their characteristics?

We start by distinguishing the related groups from the other groups by identifying IP addresses assigned by ISPs.

TABLE 4. ISPS AND THEIR NUMBER OF RELATED GROUPS

ISP	No. Related Groups
MEO Mobile	17
PT	95
NOS	101
Vodafone	87
Cabovisao	5
GVT	2
Total	307

TABLE 5. GROUPS THAT ARE NOT RELATED GROUPS

Group	No. Devices	No. Users
UTL	100	79
INESC-ID	12	9
PT Prime	12	12
Others	30	25

TABLE 6. WEBSITE ACCESSES FOR A USER OF DATASET II

Date	MacOS	Android1	Android2	Acc./day
Jan 21st	*1*	0	0	1
Jan 22nd	*2*	*1*	*1*	4
Jan 23rd	*1*	*1*	*1*	3
Jan 24th	*1*	*1*	*1*	3
Jan 25th	*2*	*1*	*1*	4
Jan 26th	*2*	*1*	*1*	4
Jan 27th	*2*	*1*	*1*	4
Jan 28th	*2*	*1*	*1*	4
Jan 29th	*2*	*1*	*1*	4
Jan 30th	*2*	*1*	*1*	4
Jan 31st	*2*	*1*	*1*	4
Feb 1st	*4*	*0*	*1*	5
Feb 2nd	*2*	*1*	*1*	4
Feb 3rd	*2*	*1*	*1*	4
Feb 4th	*4*	*2*	*2*	8
Total (devices)	31	14	15	

We did this manually as more than 95% of the accesses were made from our country (Portugal) and it was trivial to identify the ISPs. Doing this at global level may require using a database of ISPs.

We identified 6 ISPs and 26 other networks. In the ISPs there were 307 individual IP addresses, which correspond to that same number of related groups applying only Criterion 1 (see Table 4), and a total of 333 groups. Notice that considering the 307 IP addresses to be related groups is a simplification that derives from the use of a single criterion to identify related groups; in reality some of these addresses may be from other networks. These addresses from other networks might be excluded using the other criteria of Section 3.3, but this would require a larger dataset.

In relation to the groups that are not related groups, there are three main ones (see Table 5). The major are UTL, our university (outdated, now ULisboa), which is not a surprise as we asked colleagues/students to access the websites, and INESC-ID that is our research lab.

Question 7 – *Related group fingerprinting:* Is it possible to identify the devices of a related group? What is the precision and accuracy of our related group fingerprinting scheme?

To start answering this question, we analyzed data from Dataset II for a user – one of the authors – with several devices and daily accesses between January 21st and Febuary 4th, 2016. Table 6 shows some data about these accesses, which were made from three different devices. When one of the devices made an access from the home network of the user, we put that entry and those below in italics to mean that the device was recognized as being part of the related group. This allows observing that after two days three of the user's devices were recognized as being part of the related group. After that exercise, we analyzed Dataset I looking for other users of the same related group. In fact we found two other users from the same related group (which we know to be persons from the same family in this case): (1) user that made an access with an iOS (iPad) device on Sat Feb 20 2016 18:42:15 GMT; (2) user that made an access with an Android device on Sat Feb 20 2016 18:50:43 GMT.

This analysis confirms that as expected it is possible to identify devices from the same related group.

We completed this analysis by inspecting the extended Dataset I where we identified all the related groups. As most devices ended up accessing the page only once, they ended up being excluded. This way, for a total of 85 users and 101 unique devices, Table 7 summarizes the data for the 280 identified related groups. For each related group the table shows: a number we assigned to the related group (1st column); the ISP of the related group's home network (2nd); the number of devices belonging to the related group (3rd); the number of devices from this related group that made accesses from the home network (4th); the number of devices from this related group that made accesses from other networks (5th); the number of positives (P), that is equal to the value in the 4th column (6th); and the number of false negatives (FN), which are the devices from the related group that did not make accesses from the home network and therefore were not able to be assigned to the related group, but that we know to belong to the related group due to the user ID (7th).

Given the data in the table we calculated the precision of the scheme for this data that was 1, and the accuracy that was 0.999. We made one simplification that was to consider that the number of false positives (FP) is zero, thus the precision of 1; we do not have enough data to confirm this but we see no reason to believe there were false positives (devices wrongly assigned to a related group). On the contrary, the number of false negatives (FN) used was the one in the table that is pessimistic. In fact, as seen in Table 6, devices may not be added to the related group in the first contact, but they will eventually be included in the group.

5. Conclusion

We define and explore the problem of related group fingerprinting in web sites. We show that despite recent mechanisms that prevent access to browser history (a positive privacy measure), it is still possible to do device (or cross-browser) fingerprinting, at least with our main dataset of 500+ fingerprints. Related group fingerprinting leverages device fingerprinting and we have shown that it is possible to do it with high precision and accuracy. We also argue that

TABLE 7. RELATED GROUP ANALYSIS FOR DATASET I

Related Group No.	ISP	No. devices of RG	No. devices accessed home net.	No. of RG users w/accesses other nets	P	FN
1-15	MEO	1	1	0	1	0
16-17	MEO	2	1	1	1	1
18-78	NOS	1	1	0	1	0
79	NOS	1	1	1	1	0
80-86	NOS	2	1	1	1	1
87	NOS	2	1	2	1	1
88	NOS	3	1	2	1	2
89	NOS	5	1	3	1	4
90-106	NOS	2	2	0	2	0
107	NOS	2	2	0	2	0
108	NOS	3	2	1	2	1
109-110	NOS	4	4	0	4	0
111-180	PT	1	1	0	1	0
181	PT	1	1	1	1	0
182	PT	1	1	0	1	0
183	PT	1	1	0	1	0
184	PT	2	1	1	1	1
185-186	PT	2	1	1	1	1
187	PT	2	1	1	1	1
188	PT	3	1	2	1	2
189-194	PT	2	2	0	2	0
195	PT	2	2	1	2	0
196	PT	2	2	0	2	0
197	PT	3	2	1	2	1
198-199	PT	3	3	0	3	0
200	PT	4	4	0	4	0
201-242	Vodafone	1	1	0	1	0
243-244	Vodafone	1	1	0	1	0
245-247	Vodafone	2	1	1	1	1
248-250	Vodafone	3	1	2	1	2
251-252	Vodafone	5	1	3	1	4
253-267	Vodafone	2	2	0	2	0
268	Vodafone	3	2	1	2	1
269-270	Vodafone	3	3	0	3	0
271	Vodafone	3	3	1	3	0
272	Vodafone	6	3	3	3	3
273	Vodafone	11	6	4	6	5
274-278	Cabovisao	1	1	0	1	0
279	GVT	1	1	0	1	0
280	GVT	2	2	0	2	0

cross-device fingerprinting can not be done using similar features.

Acknowledgments. We warmly thank Dulce Domingos, Ibéria Medeiros, Miguel Pardal, and Nuno Santos for their feedback on previous versions of this work, and all the volunteers who contributed for the experiments. This work was partially supported by the European Commission through contract No. 610713 (PCAS) and by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013 (INESC-ID).

References

[1] N. Nikiforakis, G. Acar, and D. Saelinger, "Browse at your own risk," *IEEE Spectrum*, vol. 51, no. 8, pp. 30–35, 2014.

[2] D. Kim, "Poster: Detection and prevention of web-based device fingerprinting," in *Proceedings of the 35th IEEE Symposium on Security and Privacy*, 2014.

[3] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93–108, 2005.

[4] A. Bhargav-Spantzel, A. C. Squicciarini, S. Modi, M. Young, E. Bertino, and S. J. Elliott, "Privacy preserving multi-factor authentication with biometrics," *Journal of Computer Security*, vol. 15, no. 5, pp. 529–560, 2007.

[5] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel, "FPDetective: dusting the web for fingerprinters," in *Proceedings of the ACM SIGSAC Conference on Computer & Communications Security*, 2013, pp. 1129–1140.

[6] K. Mowery, D. Bogenreif, S. Yilek, and H. Shacham, "Fingerprinting information in javascript implementations," in *Proceedings of the Web 2.0 Security and Privacy Workshop*, 2011.

[7] L. Olejnik, C. Castelluccia, and A. Janc, "Why Johnny can't browse in peace: On the uniqueness of web browsing history patterns," in *Proceedings 5th Workshop on Hot Topics in Privacy Enhancing Technologies*, 2012.

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

[9] P. Eckersley, "How unique is your web browser?" in *Privacy Enhancing Technologies*. Springer, 2010, pp. 1–18.

[10] G. Richards, S. Lebresne, B. Burg, and J. Vitek, "An analysis of the dynamic behavior of javascript programs," in *Proceedings of the 31st ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2010, pp. 1–12.

[11] K. Mowery and H. Shacham, "Pixel perfect: Fingerprinting canvas in HTML5," in *Proceedings of the Web 2.0 Security and Privacy Workshop*, 2012.

[12] K. Boda, Á. M. Földes, G. G. Gulyás, and S. Imre, "User tracking on the web via cross-browser fingerprinting," in *Proceedings of the 16th Nordic Conference on Information Security Technology for Applications*. Springer, 2012, pp. 31–46.

[13] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, "A systematic approach to developing and evaluating website fingerprinting defenses," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 227–238.

[14] X. Pan, Y. Cao, and Y. Chen, "I do not know what you visited last summer: Protecting users from third-party web tracking with TrackingFree browser," in *Proceedings of the Network & Distributed System Security Symposium*, 2015.

[15] N. Nikiforakis, W. Joosen, and B. Livshits, "Privaricator: Deceiving fingerprinters with little white lies," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 820–830.

[16] N. Mor, O. Riva, S. Nath, and J. Kubiatowicz, "Bloom cookies: Web search personalization without user tracking," in *Proceedings of the Network & Distributed System Security Symposium*, 2015.

[17] B. Miller, "Vital signs of identity [biometrics]," *IEEE Spectrum*, vol. 31, no. 2, pp. 22–30, 1994.

[18] F. Monrose and A. Rubin, "Authentication via keystroke dynamics," in *Proceedings of the 4th ACM Conference on Computer and Communications Security*, 1997, pp. 48–56.

[19] A. A. E. Ahmed and I. Traore, "A new biometric technology based on mouse dynamics," *IEEE Transactions on Dependable and Secure Computing*, vol. 4, no. 3, pp. 165–179, 2007.

[20] H. Gilbert and H. Handschuh, "Security analysis of SHA-256 and sisters," *Selected Areas in Cryptography*, pp. 175–193, 2003.

[21] R. W. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[22] M. Vaites, "The effectiveness of a browser fingerprint as a tool for tracking," Master's thesis, Open University, UK, 2013.

[23] C. E. Shannon, "Prediction and entropy of printed english," *Bell System Technical Journal*, vol. 30, no. 1, pp. 50–64, 1951.

[24] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.