UNIVERSITY OF SOUTHAMPTON

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING

Electronic and Computer Science

Defences against Browser Fingerprinting Techniques

by

Sakchan Luangmaneerote

A thesis submitted for the degree of Doctor of Philosophy

November 1, 2018

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING

Electronics and Computer Science

Doctor of Philosophy

DEFENCES AGAINST BROWSER FINGERPRINTING TECHNIQUES

by Sakchan Luangmaneerote

When users interact with a web page, it is often straightforward to extract user data which can then be used to create a profile of that user and even to establish the identity of the user. This identity can be used to collect the behaviour of that individual user while surfing the web. In the past, users have been tracked by small files stored on their computers (e.g., web cookies, flash cookies or supercookies). These small files stored on the user's computer are designed to be a reliable mechanism for websites to recall stateful information, but can also record the user's browsing activity. If any users desire to prevent this tracking, they can select tracking-prevention features provided on all modern web browsers. However, the problem of the user privacy does not seem to be easily alleviated. The technique of browser fingerprinting has recently emerged as a novel technique which is fundamentally different from the cookie approach, in particular no files need to be stored on the user computer. The inability to observe tracking files on the user computer means that the tracking is essentially invisible and has raised considerable concern about user privacy on the Internet. This invisible tracking then can become a major problem for users who do not realise that they are being tracked by somebody without their consent.

The main inspiration for this thesis is the limited provision of existing countermeasures to assist users who wish to avoid fingerprint tracking. This research proposes a new browser fingerprinting countermeasure, called 'FP-prevention'. The primary function of FP-prevention is to obfuscate the monitoring undertaken by websites

using fingerprinting algorithms by changing the user identity on every request from the web browser to the web server. Changing the user identity using this new approach will not only assist users to avoid fingerprint tracking but also provides a significant benefit for users: when users are surfing websites, the 'look and feel' is similar to using the unmodified browser.

In part of the overall evaluation, FP-prevention is assessed on four aspects. At first, FP-prevention is measured on the web browser performance through three JavaScript benchmarks. The result suggests that FP-prevention shows trivial side effects on the web browser performance compared with an unmodified web browser. In terms of efficiency of fingerprinting prevention, FP-prevention is measured on the effectiveness of fingerprinting prevention by observing fingerprinting ID provided by three fingerprinters. The result suggests that FP-prevention is the third most effective countermeasure compared with three countermeasures. Then, FP-prevention is measured on the information paradox by observing the change of browser's attributes during a visit to the fingerprint website multiple times. The result suggests that FP-prevention shows negligible side effects on the problem of information paradox. Finally, FP-prevention is measured on the user satisfaction by conducting the survey. The result suggests that FP-prevention yields the highest score in all metrics related to the user satisfaction. With all obtained results, the research considers whether the proposed countermeasure (FP-prevention) is sufficiently robust to prevent fingerprinting tracking efficiently in combination with introducing only limited side effects to the web browsing experience.

Table of Contents

Table of Tables

Table of Tables

Table of Figures

Table of Figures

Academic Thesis: Declaration of Authorship

I, Sakchan Luangmaneerote, declare that this thesis and the work presented in it are my own and has been generated by me as the result of my own original research.

Defences against Browser Fingerprinting Techniques

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Parts of this work have been published as:
    - SAKCHAN LUANGMANEEROTE, Ed Zaluska and LESLIE CARR 2016. Survey of Existing Fingerprint Countermeasures. International Conference on Information Society 2016. Dublin, Ireland
    - SAKCHAN LUANGMANEEROTE, Ed Zaluska and LESLIE CARR 2017. Inhibiting Browser Fingerprinting and Tracking. The 3rd IEEE International Conference on Big Data Security on Cloud (BigDataSecurity 2017) Beijing, China.

Signed: .....

Date: .....

Acknowledgements

I would like to thank my supervisors Ed Zaluska and Leslie Carr their incredible support, guidance, patience, and always having time for me whenever I needed help.

My PhD would not have been possible without the generosity of Rajamangala University of Technology Isan Thailand and Thai government offering me a full scholarship for my studies at the University of Southampton.

Lastly, I would like to thank my family and friends for their supports.

Definitions and Abbreviations

PAD Model	Pleasure-Arousal-Dominance Model
WOM	Worth of Mouth
CP-Rmuti	The Department of Computer Technology at Rajamangala University of Technology Isan, Surin campus
ANOVA	Analysis of Variance
HTTP	The Hypertext Transfer Protocol
NAT	Network address translation
HTML	HyperText Markup Language
API	Application Programming Interface
DOM	The Document Object Model
CSS	Cascading Style Sheets
ECMA	The European Computer Manufacturers Association
W3C	The World Wide Web Consortium
EU	The European Union
HSTS	HTTP Strict Transport Security

Chapter 1 Introduction to Web Privacy and Browser Fingerprinting

The first chapter briefly introduces the main motivation for this thesis in Section 1.1. The research challenges will be discussed in Section 1.2. The research questions and outline of thesis are clearly described in Section 1.3 and Section 1.4 respectively. Section 1.5 formally presents the publication list of published articles based on the material from this thesis.

1.1 Motivation

Users of the World-Wide Web may possibly assume that all of their personal information remains private. In particular, they would not normally expect that the website they are currently connected to would know their full identity (unless they have explicitly provided it, of course) or know the full history of the other websites they have visited. The evolution of the web now means that such an expectation of 'Web Privacy' might be no more than an illusion. The removal of this privacy without any user consent has the potential to become a serious problem. Currently, online advertising companies are collaborating with a large number of websites to collect a user's browsing data and using this information to create a profile of the user's activities and interests. These browsing profiles can be utilised for various purposes such as targeted advertising (Johnson, 2013), web analytics (Järvinen and Karjaluoto, 2015), online purchase (Pereira et al., 2016). On the other hand, many user profiles are commercially exploited by online advertisers such as selling collected user profiles to third parties (Asay, 2012) or recording browsing behaviour without consent (Nikiforakis and Acar, 2014).

With regard to the traditional approach, many commercial companies typically use web cookies – small text files stored on the user's computer which contain details of user profiles as users are surfing the web - to track the user. Web cookies are used for various purposes by commercial companies such as allowing a site to recognize returning users, maintaining state of session or displaying targeted online advertisements. However, the

stored cookie files can also be used to track users. A user who is mindful of privacy concerns can use various features (e.g., private mode or disabled cookies) provided on modern web browsers (e.g., Firefox, Chrome or Internet Explorer) to remove or block cookie files (Aggarwal et al., 2010). The result of removing or blocking web cookies ensures that a user cannot be tracked by web cookies. Even though many web browsers have provided such features for preventing tracking by web cookies, many advertising companies have developed a new way of tracking users. For example, Flash cookies are theoretically similar to the normal cookies, but they bury the tracking information in Flash cookies by using Adobe's Flash plugin. The information would be retained in Flash cookies and then would be used to regenerate the normal cookies. In 2010, a new technique has been developed by Eckersley (2010) (called "browser fingerprinting technique"), designed to track users even if the web browser is configured to block web cookies. This is an emerging problem of user privacy.

Browser fingerprinting is an online tracking technique that is becoming popular in recent years according to the survey by Acar et al. (2013) of more than 400 websites in Alexa top million pages fingerprinting users. It can be applied to authenticate a user for banking and retail websites and to recognise users for targeted advertising. By integrating available information through JavaScript and the browser plugin, it is feasible to create a "User identification" for any online user. The characteristics of the created user identification depend on the user information accessed through a web browser. This user identification will be used to refer to the user's computer in order to collect or record the user's browsing profile. The user identification thus created will be stored on the server side without relying on the client-side stateful identifiers. For this reason, it still continually tracks users, even if they disable web cookies or use the private mode on their web browser. In addition, the fingerprinting code can be embedded in advertising banners or third party widgets (Nikiforakis et al., 2013). This ability can lead to the hidden tracking of users without their consent or knowledge.

On the user computer, most web browsers have not provided options to block or disable fingerprinting for users (Nikiforakis et al., 2014a). Thus, users have no way to protect their privacy from fingerprint tracking and furthermore many users are unaware of being tracked. They cannot distinguish which web pages are a normal page or a

fingerprinting page because the browser fingerprinting technique mainly uses JavaScript code, (normally used extensively by most websites) to fingerprint a user. Using JavaScript code turns out to be advantageous for the fingerprinting companies because the normal users cannot notice the fingerprint tracking as their web pages still work as usual. The fingerprinting code is difficult to detect as it is normally well-hidden by the fingerprinting companies (Acar et al., 2014, Nikiforakis et al., 2013). This means that the users are often completely unaware that someone is tracking them using the browser fingerprinting technique.

With a worldwide online advertising spend of more than $178 billion in 2016 (Helsloot et al., 2017), the use of the fingerprinting technique is very attractive for advertising companies. The created profiles of users may be sold to third party companies without their consent or even awareness that they have been tracked. Users cannot even be aware of what kind of (their) data is being sent to the third party companies while they are browsing the web. It can be clearly seen that this is a breach of privacy for users. Browser fingerprinting technique seems to encroach on user privacy more than other techniques because of the hidden tracking mechanism (Acar et al., 2013). As modern web browsers seem disinterested in this problem, they do not provide any features for preventing fingerprint tracking. Neglecting this issue has greatly facilitated the fingerprinting companies to exploit unwitting users while surfing the Web.

Even though there are several approaches currently proposed to inhibit fingerprint tracking, the existing approaches seem to operate inefficiently. Many researchers are attempting to come up with a new idea to inhibit fingerprint tracking. A straightforward solution attempts to stop running of the fingerprinting scripts from loading in a web browser, similar to blocking adverts. Some interesting alternatives are maintaining a blacklist of URL or detecting problematic scripts and sharing the same fingerprint between many users. However, these countermeasures can cause problems with running of a web page because many web browsers are unable to run the web page as usual after installing these countermeasures.

The research reinvents the way to prevent the fingerprint tracking, a countermeasure called FP-prevention. The idea behind FP-prevention is completely different from traditional countermeasures. FP-prevention does not use a detection stage to detect the fingerprint code. FP-prevention is designed based on the concept of changing user's attributes systematically, sufficient to prevent the fingerprint tracking and to maintain the normal state of a user experience without side effect on the web browser. The outcome of evaluating FP-prevention shows that FP-prevention is much preferable in terms of user experience and reducing the problems of information paradox. However, the weakness of FP-prevention is that it is unable to prevent the fingerprint tracking effectively, compared with other countermeasures. Finally, this research proves that changing attributes systematically can prevent the fingerprint tracking, reduce the probability of a web browser standing out and maintain the normal state of a user experience without relying on fingerprinting detection.

1.2 Research Challenges

In order to assist users to avoid fingerprint tracking, the proposed countermeasure needs to provide assistance to users to deal with the fingerprint tracking along with considering the impact on the user browsing experience.

After an extensive review of the current existing literature in Chapter 2 and Chapter 3, it is found that almost all fingerprint countermeasures are impracticable and ineffective in assisting users to avoid fingerprint tracking. A new countermeasure is required to assist the user to avoid fingerprint tracking without adversely impacting the user browsing experience.

1.3 Research question and contribution

The research is into the critical issue of protecting user privacy while users are surfing the web. The literature review reveals that there are currently no countermeasures capable of preventing browser fingerprinting effectively. In addition, almost all countermeasures attempt to handle the browser attributes without concern about the consequence of changing the browser attributes. This action causes problems for the user

browsing experience. Therefore, the ideal fingerprint countermeasure should combine effective fingerprinting prevention and insignificant side effects to the web browser.

This research question will be the main contribution of the PhD thesis:

> *How can randomized attributes be introduced into a web browser in order to prevent browser fingerprinting with a minimal impact on the browsing experience?*

1.4 Outline of the thesis

This research consists of nine chapters. Figure 1-1 shows the relationship between each chapter.

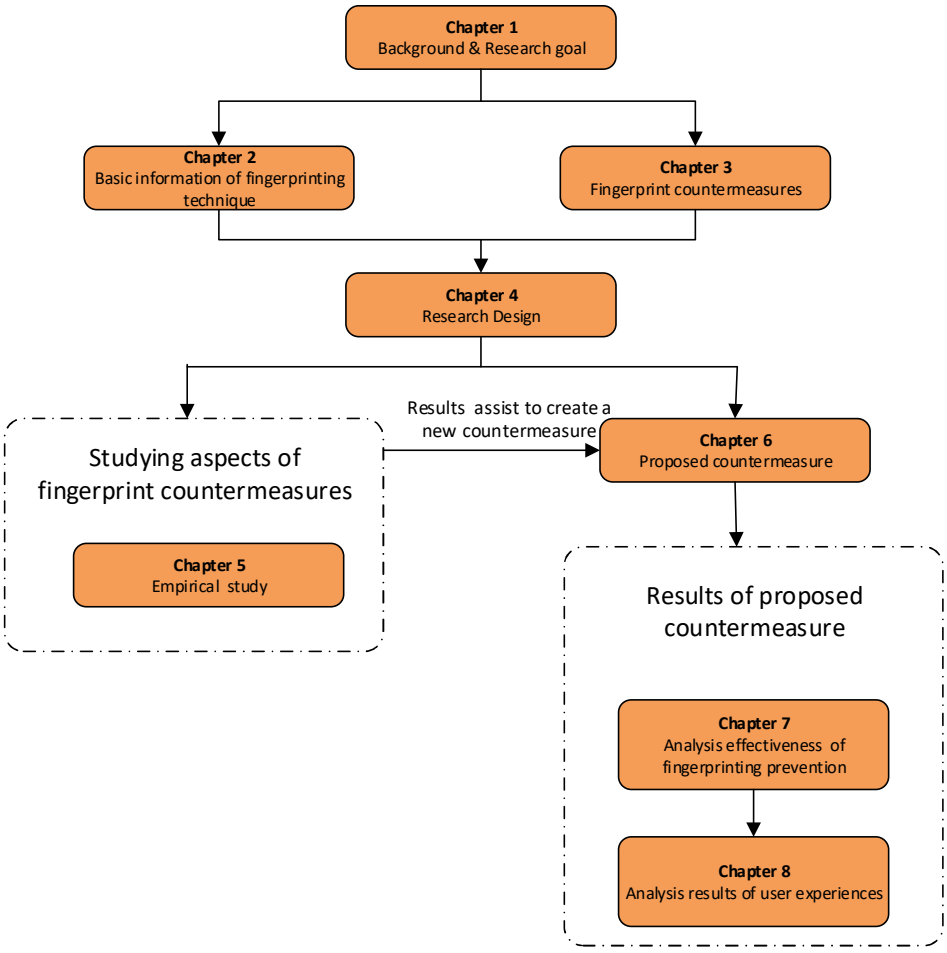


Figure 1-1 diagram represents the relationship between chapters

Chapter 1 discusses the problems of user privacy as a result of developments of tracking technologies. The browser fingerprinting technique is one tracking technology which is difficult to inhibit. The existing countermeasures appear inefficient in dealing with the fingerprinting prevention and create unwanted side-effects to the web browser.

Chapter 2 provides an in-depth overview of the browser fingerprinting techniques associated with this research. The chapter starts with an overview of the current tracking technologies and the basic concept of the browser fingerprinting technique. Afterwards, this chapter provides types of browser fingerprinting technique and attributes used for fingerprinting. Finally, this chapter includes information of how many companies have adopted the browser fingerprinting technique for tracking Internet users.

Chapter 3 lists a number of fingerprint countermeasures used for inhibiting the fingerprint tracking. This begins with a discussion of the possible ways to inhibit fingerprint tracking. This chapter attempts to classify types of fingerprinting prevention in order to determine the possibility of further development. This chapter provides additional information about problems of user experience and the information paradox. Inattention to problems of user experience and the information paradox means a fingerprinting countermeasure cannot prevent fingerprint tracking effectively.

Chapter 4 clarifies the procedural plan for how this research will be accomplished. This chapter begins with the overview of research design, consisting of problem identification, solution design and evaluation. Lastly, this chapter attempts to explain all tasks to be conducted in this research.

Chapter 5 explains the empirical study of fingerprinting countermeasures. This chapter discusses how each fingerprinting countermeasure works and what are the advantages and disadvantages of each countermeasure. The obtained results are then used when creating a new countermeasure in the next chapter.

Chapter 6 explains details of the proposed new countermeasure. This chapter begins with a discussion of the problems of the existing countermeasures and proposes to address the limitations of existing countermeasures with a new countermeasure. Details of the proposed new countermeasure are discussed and described.

Chapter 7 provides the methodology of validating the effectiveness of fingerprinting prevention. The proposed new countermeasure is validated by comparing with other existing countermeasures to determine whether it performs effectively or not.

Chapter 8 describes the methodology of conducting the survey. The results of the survey demonstrates whether the proposed countermeasure can perform effectively as expected (showing insignificant web browser side-effects) or not, compared with other countermeasures. This chapter explains details of all the tasks of obtaining the data in order to validate the level of user satisfaction.

Chapter 9 discusses the results associated with the exploration of this research in addition to the limitations of the study. Finally, this chapter provides the conclusion of this research study in addition to suggesting possible directions of future research study.

1.5 Publication list

The following peer-reviewed conference papers using the work of this thesis have been published.

SAKCHAN LUANGMANEEROTE, Ed Zaluska and LESLIE CARR 2016. Survey of Existing Fingerprint Countermeasures. International Conference on Information Society 2016. Dublin, Ireland

SAKCHAN LUANGMANEEROTE, Ed Zaluska and LESLIE CARR 2017. Inhibiting Browser Fingerprinting and Tracking. The 3rd IEEE International Conference on Big Data Security on Cloud (BigDataSecurity 2017) Beijing, China.

(This page intentionally left blank)

Chapter 2 Literature Review

This chapter discusses the general background of browser fingerprinting relevant to the problem introduced in Chapter 1. This chapter begins with Section 2.1 which presents the advantages and disadvantages of web tracking. Then, the overview of the current tracking technologies will be discussed in Section 2.2. Section 2.3 clarifies the concept of web browser fingerprinting, followed by the concept of identifying information in Section 2.4. The browser fingerprinting methods will be elucidated and then the overall process of the data flows of browser fingerprinting technique will be explained in Section 2.6. Section 2.7 explains how fingerprinting code providers use the browser fingerprinting technique on the Web. Finally, a summary of the chapter is provided in Section 2.8.

2.1 Advantages and disadvantages of web tracking

Web tracking is recognised as web log analysis, web log file analysis and web logging (Fourie and Bothma, 2007). Web tracking can offer valuable information to develop a website. It is a method of collecting activities of the Internet users while browsing web pages. The data collected and analysed from the Internet users will be turned into the user profile and then will be used for various purposes such as showing online advertisements, monitoring traffic or improving user experience (Mowery et al., 2011b). The generated user profiles immensely benefit the Internet users to navigate the web. For example, Internet users no longer need to login to a web page whenever they go back to the same page after the first login. In the case of law enforcement, web tracking can be used to assist the government for seeking someone's criminal record via traces left on Internet activities (Federrath, 2005). Moreover, it also supports website owners to assess the number of visitors who are interested in their websites. This helps indicate to a website owner which web page is attractive for visitors and which web pages should be improved (Kent et al., 2011, Rastogi and Khan, 2015, Evans, 2009, Phippen et al., 2004). For these reasons, many advertising companies continually attempt to develop many trackers (e.g., web cookies, Flash cookies or supercookies) for different purposes.

However, web tracking is a cause of widespread concern for privacy of user information. Nikiforakis et al. (2014a) have discovered that browser profiles of users are being misused by commercial organisations and government snoops for various reasons (e.g. advertising, analytics, social network, selling user profiles to other companies and more). According to Turow et al. (2009), they conducted a survey of how the user thinks about tracking. The result showed 87% of respondents were uneasy with the idea of allowing anyone to track them without consent.

2.2 Overview of the current tracking technologies

Tracking technologies can be classified into two types: stateful and stateless (Vanitha and Arthibala). Stateful tracking is a technique that relies only on a unique string placed on the user's computer (such as cookies, Flash cookies and HTML local storage) and is sent with every sequence request (Bau et al., 2013). The use of cookies is ubiquitous in early 2006 by Krishnamurthy and Wills (2006). These trackers are used to maintain a session on the Hypertext Transfer Protocol (HTTP) protocol (which was deliberately designed to be a stateless protocol) and to track users on the Internet (Donaldson, 2008, Lambrecht and Tucker, 2013). Typically, web cookies are stored for a limited time on the user's computer, but other types of cookie, such as supercookie, HSTS super cookies and Flash cookies, are designed to be stored permanently on the user's computer. Many types of research have demonstrated an abuse of stateful tracking - for instance, using flash cookies to regenerate cookies by using HTML5 localStorage and Etags to restore cookies[1] (Sorensen, 2013, Englehardt and Narayanan, 2016b). Usually, these cookies settle in part of the user's computer where it is difficult to delete them, including Flash, IE Local Storage and Local Shared objects. However, many modern web browsers are starting to create methods to remove these cookies, but these methods are not automatic and require user awareness[2].

Stateless tracking, or fingerprinting technique, was recently designed to create a user identification through a web browser (Eckersley, 2010) or traffic networks (e.g. IP address or HTTP header) (Yen et al., 2012a). The stateless tracking exploited weaknesses of web

[1] http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1898390
[2] https://blog.chromium.org/2011/04/providing-transparency-and-controls-for.html

technologies, normally designed to customize a web page (e.g. HTML5, JavaScript, CSS or browser plugins), for creating a user identification. Typically, these web technologies provide application programming interface (API) and variables (e.g. JavaScript object) which can easily obtain the user data. The straightforward access to the user data becomes increasingly beneficial to the fingerprinting technique in creating the unique tracker, but it is worse for a user who desires to protect their data. Moreover, stateless tracking is not necessary to store any identifiers on the user computer. Unlike stateful tracking, stateless tracking retains an identifier to the fingerprinting server. The stored identifier on the fingerprinting server can collect a number of websites or web pages used by a user to a database. For this reason, users are fully unaware that they are being tracked because they cannot find any identifiers (e.g. web cookies) on their computer. Stateless tracking is hidden tracking. It is extremely difficult to detect and prevent (Acar et al., 2014). Furthermore, the browser vendors are not provided any ways to prevent the fingerprint tracking, as opposed to web cookies. Therefore, this leads to problems of data protection and data privacy.

2.3 The concept of web browser fingerprinting

Device fingerprinting is a set of user's system attributes which are typically concatenated in the form of a string. This combination of user's attributes is particularly created to be unique with a high probability and can perform as a device identifier (Nikiforakis et al., 2014b). The real motivation behind this technique is similar to human fingerprinting. The general concept of device fingerprinting is that the diversity of user's attributes on the computer contributes to the creation of a unique identity similar to human fingerprinting. The process to create the unique tracker derives from the access to different information on the user device. The obtained information from the access to the device will be concatenated into a string and then put into the Hash function. The output of the Hash function is a Hash ID which will be used as an identifier. This Hash ID will be used to refer to the identity of the user's device when a user browses the web. This technique can be implemented in various ways for tracking the user (e.g. browser, mobile fingerprinting or other devices). (Hupperich et al., 2015, Eubank et al., 2013, Nakibly et al., 2015)

Similarly, this technique can be applied to a web browser. It is then called '*browser fingerprinting*'. A web browser (usually just called browser) is a software application that Internet users need to use in presenting, retrieving and traversing information resources on the World Wide Web. Currently, there are many different web browsers (e.g. Internet Explorer, Chrome or Firefox) available for use on the Internet. With the rapid development of web browsers, many web browsers have security holes (e.g. cross-site scripting or man-in-the middle attack) that can be compromised by the attacker. Moreover, technologies or scripts used on the web browser, such as CSS, DOM, JavaScript and plugins, are normally designed to enrich the user's experience, but they appear to facilitate a typical web server to easily gain basic information on a user via the web browser (Sinha and Choudhary, 2014). The straightforward access to user information to assist the concept of device fingerprinting technique can be used in practice − it can create a unique tracker easily through the use of a web browser.

Using the browser fingerprinting technique, a user can be tracked consistently because a web browser would send the basic information to the website with every HTTP request such as language, browser version, and operating system (Nikiforakis et al., 2013). JavaScript usually used on the website would assist the browser fingerprinting technique to obtain the additional information of a user such as screen resolution, list of installed plugins, or timezone. When combined with each piece of information, it is sufficient to create the unique fingerprint tracker. For example, List of system fonts is one of the high-entropy-attribute (13.9 bits of entropy). It means that 1 in 15286 web browsers visiting website show the same list of system fonts**.** By integrating this attribute with other attributes, the combination could be unique or shared with only a small subset of web browsers.

The browser fingerprinting technique will not depend only on the IP address. The fingerprinting algorithm could distinguish users even if users are behind a Network address translation (NAT). Any effects on the user computer, such as updating the operating system, the browser or plugins, using the new browser, should not affect errors in fingerprinting identification if this technique provides an algorithm to learn a change of attributes (Flood and Karlsson, 2012). With the difficulty to inhibit data extraction, users can be tracked despite using privacy mode on a web browser (Broenink, 2012a).

2.3.1 Stability of browser fingerprinting

With the nature of user identification created by the browser fingerprinting technique, Eckersley (2010) had shown that fingerprinting will change as time goes by. The change of fingerprint is a result of automatic update of web browser version, updated fonts or plugins and so on. Thus, the fingerprinted user would gradually change over time. Ideally, the fingerprint ID should not be changed after being fingerprinted. In reality, the stability of browser fingerprinting technique in practice means that any changes to user information on the user computer (e.g. updating the web browser or updating the operating system) can be distinguished by the browser fingerprinting technique (Boda et al., 2012a, Alaca and van Oorschot, 2016). Therefore, even though any attribute values can be changed, the good browser fingerprinting technique still distinguishes changes of attributes - stability of fingerprinting is not compromised. For example, Eckersley (2010) developed a sample heuristic algorithm to predict the change of version of a web browser. Alterations of upgraded version of a previously noticed browser on the user computer can be identified by using his algorithm with over 99% accuracy.

In order to maintain the stability of fingerprinting, many fingerprinters would choose attributes which are not frequently changed such as operating system, cookie support, language, Do Not Track and so on. This manner can mitigate the problem of stability of fingerprinting along with improvements in the fingerprinting algorithm to learn the gradual change of attributes.

2.3.2 Diversity of fingerprint tracker

The diversity of fingerprint tracker is absolutely essential for the browser fingerprinting technique to identify the user's computer. The browser fingerprinting technique should produce different fingerprint trackers after a user is fingerprinted. In terms of the browser fingerprinting technique, a user computer has sufficient diversity to create a unique tracker. Typically, a user computer which has a high diversity of attributes will increase the probability of creating a unique fingerprint. On the contrary, if any computer has a low diversity of user attributes, it increases the probability of creating a non-unique fingerprint (the same fingerprinting ID). The same fingerprinting ID thus

created will cause trouble for the fingerprinting technique because the fingerprinting algorithm cannot distinguish the user computer.

Losing too much entropy value can affect the diversity of fingerprinting. The entropy value is a significant factor in increasing the diversity of fingerprint tracker, generating the unique fingerprinting ID. For example, Eckersley (2010) found that a list of plugins has 15.4 bits of entropy. It means that 1 in 43237 web browsers visiting his website[3] show the same fingerprinting ID. He used remaining attributes which are sufficient to create the unique fingerprinting ID. Another example, if the browser fingerprinting technique can obtain only Timezone. The Timezone is 3.04 bits of entropy, according to Table 2-1. It can be clearly seen that the browser fingerprinting technique has a probability of 1 in 8 of creating the same fingerprinting ID. Therefore, entropy value of attributes plays a crucial role in generating the unique fingerprinting ID.

Lastly, the diversity of fingerprinting ID can be decreased if the browser fingerprinting technique can access few attributes. If the browser fingerprinting technique can access few attributes through the web browser, the fingerprinting technique will have a very limited choice to choose suitable attributes for creating the unique fingerprinting ID. Conversely, if the browser fingerprinting technique could access the user's entire attributes, the probability of creating a unique tracker will be increased as it can select suitable attributes for fingerprinting from a wide range.

2.3.3 **Advantages and disadvantages of fingerprinting**

Recognizing the user's identity has its advantages and disadvantages. To consider advantages, it can be utilised effectively for web analytics (Roesner et al., 2012), or can be prevented with a fraud online such as identity theft or credit card fraud (Caldera et al., 2016, Nikiforakis et al., 2013). Institutions or banks that need high levels of security may implement the fingerprinting technique to support an additional level of authentication. For example, the browser fingerprinting technique is applied in combination with the anti-fraud payment system for authenticating the user's login status. The user data which is

[3] https://panopticlick.eff.org/

directly obtained through the web browser, such as IP address, User-agent, screen resolution, timezone, system language or CPU characteristic, will be used to assist the anti-fraud payment system for checking the status of the user. If the obtained user's information is different from the previous login, the anti-fraud payment system will reject a login as the user is suspected to be fraudulent. With regard to the disadvantages of browser fingerprinting, it can be used to regenerate a web cookie (Acar et al., 2014), spread malware (Egele et al., 2009) or track users (Eckersley, 2010).

2.3.4 Implementation of fingerprinting technique

Many surveys have been conducted to find out whether the fingerprinting technique has been applied in the real world. According to the Figure 2-1, Nikiforakis et al. (2013) had discovered that many popular websites (e.g. Skype.com) and unpopular websites on the Internet have used the fingerprinting technique for various reasons. For instance, the browser fingerprinting technique has been implemented by pornographic sites with the purpose of detecting the shared or stolen credentials of paying members. Another example is dating sites. They use the fingerprinting technique to verify malicious attackers who create profiles for social-engineering purposes.

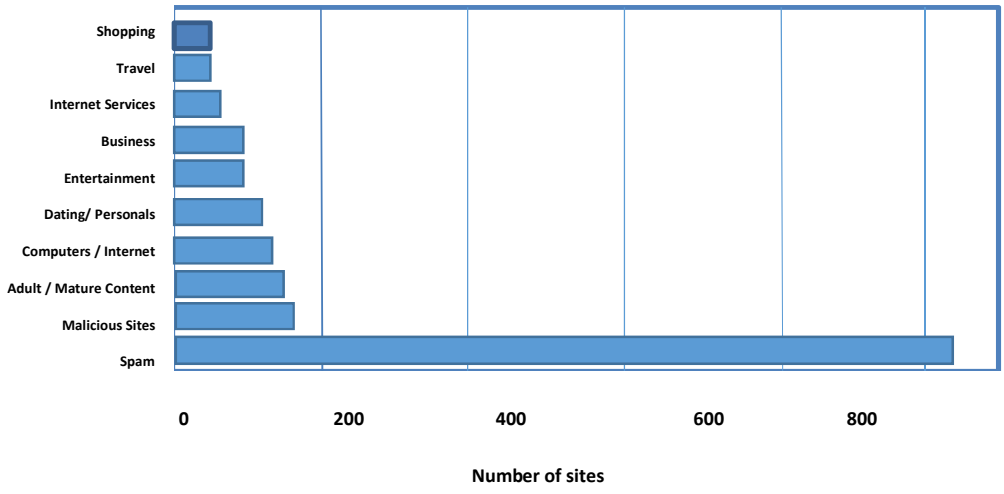


Figure 2-1 the top 10 categories of sites using fingerprinting techniques by Nikiforakis et al. (2013)

In addition, the browser fingerprinting technique is used most by spam sites and malicious sites, followed by adult content, computer, dating, entertainment, Internet services, travel and shopping sites respectively.

2.4 Identifying information

The concept of browser fingerprinting technique is associated with the idea of identifying information[4]. The idea of identifying information is to use available information to distinguish an individual identity. It is extremely useful to tell who unknown people are. Generally, nature always creates a person as a unique personal identity. Considering a single person, it comprises various attributes (e.g. hair, blood, skin colour, face or voice). These attributes can be utilized to identify a person because the diversity of attributes on the human body is sufficiently distinct to create the unique identification. Currently, research has invented various ways of identifying a single person, such as DNA (Vos et al., 1995), voice analysis (Hong and Jain, 1998), handwriting analysis (Zuo et al., 2002), iris scan (Bal and Acharya, 2011) and fingerprint (Isenor and Zaky, 1986), to implement for various purposes. Using different attributes on the human body to identify a person, there are significant benefits to track or identify somebody who lives on this planet. For example, the police could identify escaped criminals by using traces (e.g. footprint, blood, or fingerprint) left at a crime scene. These techniques can be used to arrest a crime suspect.

The attribute data is extremely important for identifying someone. To regard the identification of a person, assuming the data has only ZIP code, identifying cannot predict an unknown person accurately as the obtained information (zip code) is insufficient. If the obtained data has the date of birth, gender, and ZIP code, identifying could be possible to determine the unknown person. Therefore, identifying someone depends on obtaining sufficient information.

[4] https://www.eff.org/deeplinks/2010/01/primer-information-theory-and-privacy

2.4.1 **Entropy**

In information entropy, there is a method to measure how near a fact comes to disclosing someone's identity uniquely. This method is called 'entropy'. The entropy value can represent the degree of uncertainty of information content. If a measured piece of information is difficult to guess by chance (high degree of uncertainty), the entropy value will be high. On the other hand, if a measured piece of information is easy to guess by chance (low degree of uncertainty), the entropy value will be low. The entropy value will always be represented in the unit of bits. Moreover, entropy is close to the number of different possibilities. For instance, two possibilities are equal to 1 bit of entropy. Likewise, four possibilities are equal to 2 bits of entropy. Another example: identifying someone as male or female uses at least 1 bit of entropy to identify a person uniquely (2 possibilities). Entropy value is inevitably associated with the number of possibilities. The entropy of each piece of information can be measured by using the formula below.

$$\Delta s = -\log_2 \Pr(X=x) \quad (1.1)$$

Assuming that Web and Internet science research group at the University of Southampton has about 100 PhD students. Thus, finding someone who is studying in this group will use the formula below.

$$\Delta s = -\log_2 \Pr (1/100) = 6.64 \text{ bits of information.}$$

Knowing an additional information of unknown student assists to identify a student such as knowing the date of birth.

$$\Delta s = -\log_2 \Pr(\text{Birthday= 5 of January}) = -\log_2 \Pr(1/365) = 8.51 \text{ bits of information.}$$

Another example, the day of the week:

$$\Delta s = -\log_2 \Pr(\text{the day of the week =Monday}) = -\log_2 \Pr(1/7) = 2.80 \text{ bits of information.}$$

Note that recognising the day of the week cannot assist date of birth in learning anything new from available data. Sometimes, obtained data cannot assist in identifying individual identity. It can be seen that recognizing a number of students and date of birth

will use 6.64+8.51=15.15 which is probably sufficient for identifying an unknown student who is studying on Web and Internet science research group at University of Southampton. However, someone else who is studying on Web and Internet science research group at University of Southampton may share the same identity. If this technique also recognises a gender, it uses 16.15 bits (adding 1 bit) which decreases the probability of being mistaken for someone else.

2.4.2 **Implementation to the web browser**

Many attributes of the user's computer are easily accessible by the web server through a web browser (Wadkar et al., 2014). Both browserleaks[5] and browserspy[6] show clearly that they can easily gain access to the user data via a web browser as users are browsing the web. For example, User-agent comprises many attributes inside of it (e.g. operating system, versions of the web browser or a web browser engine) and always notifies the network protocol in order to recognise what kind of information used between the user computer and web server. A form of User-agent is displayed below.

Mozilla/5.5 (Windows; U; Windows NT 5.0; en-GB; rv:1.7.1.3) Gecko/20080624 Firefox/2.0.0.7

With regard to the information above, many attributes have been contained in the User-agent. It appears that it is extremely useful to learn personal user data. According to Eckersley (2010) and Boda et al. (2012a) in Table 2-1, each attribute on the user computer has a different entropy value. They established that a list of fonts, a list of plugins and User-agent have a high entropy value. For example, User-agent header provided 10.5 bits of entropy, according to an experiment of Eckersley (2010); it means that 1 in 1500 web browsers visiting his website show the same User-agent header**.** Therefore, it is highly possible that a user's computer can be identified using attributes of the user's computer.

[5]://www.browserleaks.com
[6]http://browserspy.dk/

To consider the use of low entropy attributes on the user's computer in order to fingerprint, many fingerprinting companies can freely choose these attributes depending on the purpose of their business. Increasing the number of low entropy attributes assists to easily create the unique fingerprint and also supports the learning of more user data. In case that the browser fingerprinting technique is unable to access some high entropy attribute, low entropy attributes can be used to assist the browser fingerprinting technique for creating the fingerprinting ID by using the remaining attributes. Therefore, both low entropy and high entropy contribute to the diversity of fingerprinting ID.

Table 2-1 the different attributes with different entropy

	(Eckersley, 2010)	(Broenink, 2012)	(Boda et al., 2012)	(Yen et al., 2012)
Attributes	**Entropy(bits)**	**Entropy(bits)**	**Entropy (bits)**	**Entropy (bits)**
HTTP User-agent	10	6.31	8.1	11.59
HTTP Accept	6.09	2.14		
HTTP Accept-language		4.25		
HTTP Accept-encoding		1.84		
HTTP Accept-charset		1.83		
HTTP Concetion		0.37		
HTTP DO Not Track		0.57		
Plugin Version	15.4	6.25		
Font list (ALL)	13.9	6.23	8.57	
Screen resolution	4.83	4.58		
Supercookies Partial	2.12			
JavaScript Enabled		0.79		
JavaScript version		2.09		
Operating system		2.13		
Language		2.24		
Java Support		0.95		
TimeZone	3.04	1.82	2.22	
Cookies Enabled	0.35			
A number of visiting users	**470,161**	**1,124**	**989**	**1,771,907**

To calculate the entropy, each piece of measured information (Δ**s)** can be calculated using Shannon's entropy formula below, p_i is the probability of occurrence of i value.

$$H = -\sum_{i=1}^{n} p_i \times log_2 p_i \qquad (1.2)$$

2.5 Browser fingerprinting methods

Currently, many browser fingerprinting techniques are established for different purposes and different methods. Acar et al. (2013) had reasonably proposed that browser fingerprinting methods can be classified into four categories. Some of categories are already being used on the Internet, but other categories are at the stage of theoretical development and have not been implemented in practical systems.

2.5.1 **JavaScript-based**

Currently, most websites use JavaScript to improve the desired web page to extend functionality. Usually, JavaScript can be used to assist browser developers in creating web pages and improve the user experience (Richards et al., 2010). JavaScript is operated on the client-side, and the scripts are embedded into the HTML code of a web page. When a web browser is loading a web page, the script will be downloaded into a web browser and then is interpreted for running. A result of running depends on an assigned task which normally uses DOM model that provides plentiful objects of prewritten functionalities. JavaScript was standardised by the European Computer Manufacturers Association (ECMA) in 1997 as ECMAScript (Ecma, 1999). For security reasons, a developer cannot use JavaScript for reading or writing file exceptions of cookies (McGrath, 2013). Using JavaScript is a convenient way to obtain basic information about the user through JavaScript objects.

The accessed JavaScript object can be employed to assist the browser fingerprinting technique for creating the unique user identification, according to a report by Nikiforakis et al. (2014a) that more than 400 of the million most popular websites use only JavaScript for fingerprinting. Many JavaScript objects can be used to disclose real information of the user computer such as navigator (the object containing the basic information of client) and screen. This method was utilized by Mayer (2009) who was the first person to report the identification of a user computer by using JavaScript objects. He used navigator, screen, navigator.plugins and navigator.mimeTypes to concatenate into one string and he then inserted the concatenated string into the Hash function. The output of the Hash function is Hash ID, similar to a user identity. He could uniquely identify more than 96% of all the

browsers in his study with 1,328 participants. This means that JavaScript objects are provided with a straightforward mechanism to create the user identification.

After the approach of Mayer (2009) was proposed to the public, this method was developed further by Eckersley (2010). He had established the Panopticlick project[7] for investigating possible mechanisms to fingerprint the user computer in practice. He used eight attributes, namely User-agent, HTTP ACCEPT header, Cookies enabled, Screen resolution, Timezone, Browser plugins, System fonts and Supercookies, to fingerprint the user computer with more than 94.2% accuracy. His experiment can distinguish a user at 18.1 bits of entropy with a sample of 470,161 browsers. Almost all attributes of his experiment are found through the use of JavaScript. In addition, he developed a heuristic algorithm to predict the evolved version of fingerprinted browser seen previously. His algorithm can uniquely identify the user computer even though users will update version of a web browser on their computers.

Subsequently, Mowery and Shacham (2012) implemented a new fingerprinting technique using JavaScript instructions called canvas fingerprinting. The canvas in HTML5 will be used for this method to write and read an image while rendering the web page. With the value of a retrieved image, this shows the unique characteristic of the user operating system which can then be utilised to fingerprint a web browser. Moreover, Olejnik et al. (2012) used browsing history to implement fingerprinting, which can detect 69% of browsers in a study. According to Nikiforakis et al. (2012), they revealed that various providers, such as Google, use JavaScript to extract the user history from a web browser for different purposes. Similar to the experiment of Broenink (2012a), using the browser vendor, the installed fonts and plugin versions, he could identify users uniquely with an entropy of 8.47 bits despite updating browser version and plugins, his data set of 1,124 visitors.

In order to improve the stability of fingerprinting, Boda et al. (2012b) introduced a new method to fingerprint a cross-browser. They implemented first two octets of the IP

[7] https://panopticlick.eff.org/

address along with various attributes of the JavaScript to track user across a web browser. In addition, they created a new way to obtain the list of fonts without using plugins. Their methods can significantly improve the way to track the user computer rather than relying only on the browser plugins. Other research had proposed a new fingerprinting technique by observing differentiation between JavaScript engines (Mowery et al., 2011b). However, there is no significant result in success of this method for it to be implemented for tracking in the real Internet. As Olejnik et al. (2015) introduced a new method to fingerprint a user using navigator.getBattery() in JavaScript, their method utilized API of the battery Status to fingerprint.

2.5.2 **Plugin-based**

A browser plugin is an additional piece of software which supports a web browser by providing a specific additional function. For example, a Flash plugin is used for displaying video through a web browser. Moreover, Plugin APIs (Application Programming Interfaces) can be employed to reveal user data, such as CPU speed, list of fonts installed on the user computer, and CPU manufacturer, which is extremely helpful for fingerprinting the user computer (Mowery et al., 2011a). Currently, using plugins to fingerprint the user computer is an efficient way as many modern web browsers have provided plugins to enhance the user experience. In addition, Eckersley (2010) had found that if any web browser installs Java and Flash, he could use either Java or Flash plugin to access system fonts and then bring obtained system fonts to combine with other attributes for creating a fingerprinting ID. In addition, Nikiforakis et al. (2013) had disclosed that many advertising companies had used Flash plugin to circumvent defence of fingerprinting identification even though a user attempted to prevent it by using HTTP proxies. However, the latest investigation using Java plugin seems unpopular as fingerprinting companies appear to be uninterested in using it due to unpopular use of Java (Nikiforakis et al., 2013).

2.5.3 **Extension-based**

This method uses side effects of using the browser extension to fingerprint the user computer. Typically, a browser extension is an additional piece of software that provides extra functionality to a web browser (Barth et al., 2010). A browser extension is entirely

different to a plugin because it can create toolbars and run an automatic process, but a plugin is unable to function similar to a browser extension. Normally, a browser extension is created by mainly using web technologies such as HTML, JavaScript and CSS in order to execute an specific task in the web browser. Even though the browser extension is not enumerable via JavaScript, it can be fingerprinted by monitoring the side-effects. Research by Mowery et al. (2011) discovered that it is possible to analyse the custom whitelist from the NoScript extension by sending the request from domains and checking whether the script is executed or not. If the script is executed – it indicates that URLs are allowed in the whitelist, and the list of allowed URLs will be used as with one attribute for fingerprinting. However, there is no concrete result showing that this method is used in practice. In addition, a user who uses some spoofing extensions could be detected easier because fingerprinter can notice the inconsistency in the data reported (Nikiforakis et al., 2013). For example, a device pretends to be an iPhone by modifying the HTTP header, but if the fingerprinter inspects that device spoofed has Adobe Flash support, the fingerprinter can hardly believe that this is a real information of user computer because Apple does not support Adobe Flash. Therefore, using extensions to prevent tracking may make the web browser stand out, and fingerprinter may recognise unusual occurrence and then find another way to fingerprint the user.

2.5.4 **Header-based and server-side**

The side-channel information generally represents a primary source of information for traffic analysis (Danezis and Clayton, 2007, Gong et al., 2010). For instance, keystroke intervals can be analysed to predict the password that may be in use (Song et al., 2001) or packet size and rate may be monitored to disclose the presence or absence of Skype traffic (Bonfiglio et al., 2007). A number of publications had used traffic analysis to implement web fingerprinting - for example packet ordering information (Lu et al., 2010), clock skew (Kohno et al., 2005), analysing the TCP timestamps (Kohno et al., 2005) and User-agent (Yen et al., 2012b). Yen et al. (2012b) had utilised IP addresses and the User-agent header and suggest that this might be sufficient to track users. However, these techniques appear to

be less accurate than JavaScript-based fingerprinting (Acar et al., 2013). These methods are not necessary to run code on the user's device.

2.6 The overall process of the data flows of browser fingerprinting technique

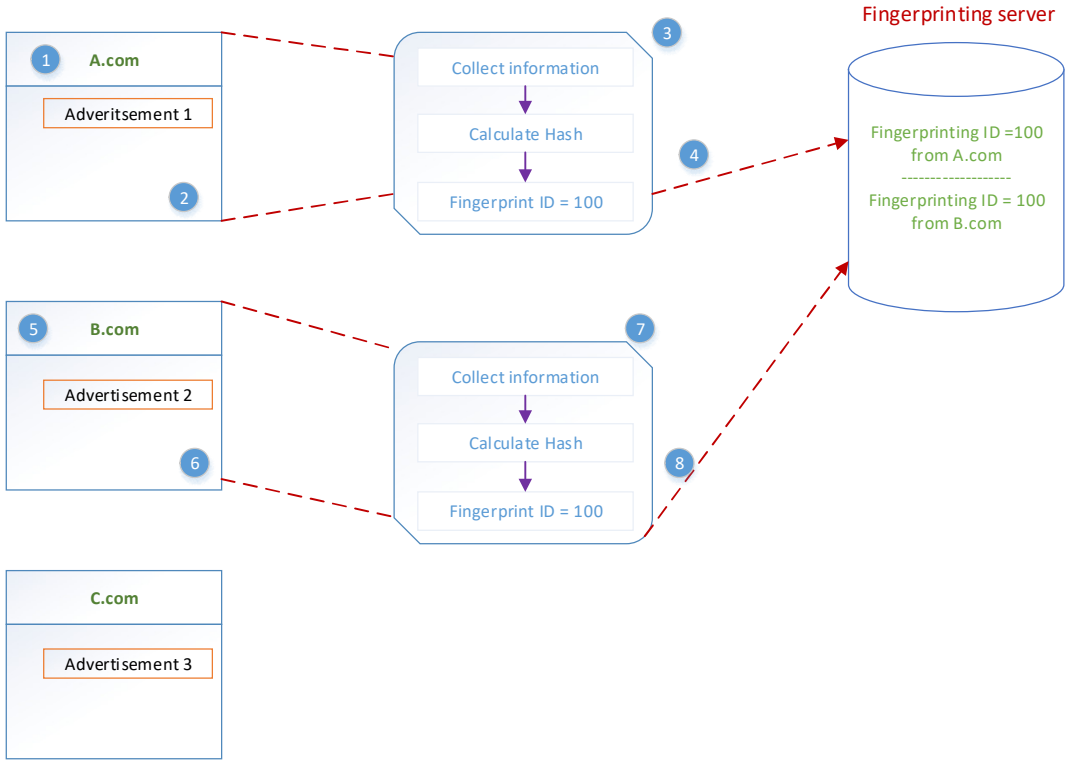


Figure 2-2 the overall process of data flows used for tracking user with fingerprinting technique

In order to illustrate the basic concept behind the browser fingerprinting technique, data flows used in the technique will be shown in this section as Figure 2-2.

1. A1 user visits the website of a.com (J1)

2. J1 will load advertisement 1 from Third-partyTracker-1.com (T1)

3. T1 will collect user data through the web browser. The obtained information will be concatenated and then put to Hash function in order to generate a fingerprinting ID.

4. T1 will send the created fingerprinting ID along with the URL of the visited website and user information to the fingerprinting server which builds up a user profile, including that the user has visited a.com.

5. User visits another site b.com which includes the same advertisement, Third-partyTracker-1.com.

6. T1 will load advertisement 2 from Third-partyTracker-1.com (T1)

7. T1 will collect user data from the web browser through b.com. Then it will make a fingerprinting ID

8. T1 will update the user profile that the user had visited b.com to the server.

However, fingerprinters attempt to improve a new way to tracking a user rather than using the traditional method (using only fingerprinting ID) in order to track a user as long as possible, more details about the smart fingerprint in Appendix H.

2.7 Fingerprinting Code Providers

The results of the Panopticlick project[8] have drawn attention to how many companies had applied the browser fingerprinting technique to track users. To investigate which fingerprinting attributes are used to track the user, they will be listed in this section, in Table 2-2. The Bluecava[9] and Iovation[10] fingerprinting companies are selected due to the high rank on the search engine in the survey of Mayer and Mitchell (2012). In part of open source software, Petportal (Boda et al., 2012a), FingerprintJS[11] and Fybridjs[12] are selected in order to compare with the fingerprinting companies, Bluecava and Iovation. Based on comprehension of how many attributes are used to fingerprint, it assists research in finding an excellent approach to inhibit fingerprinting tracking.

[8] https://panopticlick.eff.org/
[9] http://bluecava.com/
[10] https://www.iovation.com/
[11] https://github.com/Valve/fingerprintjs
[12] https://github.com/ammosh/fybridjs

- **Panopticlick:** Panopticlick is a research project intended to reveal online tracking and test the efficiency of privacy. When a user visits this site, the user's information would be combined in order to fingerprint the user. Panopticlick does not return a value for the fingerprint ID (the result returned is, for example, "Your browser fingerprint appears to be unique among the 136,250 tested so far").

- **Bluecava:** Bluecava is a technology company that offers users information on Internet usage to advertisers and advertising networks in order to increase their business opportunities. Bluecava has provided an opt-out page for users who need to be "untracked" - this page generates a fingerprint ID e.g. 8774d281-a71d-4186-bfff-b4151f350c5d.

- **Iovation:** is a company providing online fraud prevention. This company has implemented the browser fingerprinting technique to assist customers in identifying malicious attackers to their websites.

- **PetPortal:** This website has been developed by Boda et al. (2012) and was inspired by the Panopticlick project[13]. They attempt to enhance a fingerprint technique for stability purposes by using different attributes from the Panopticlick project.

- **FingerprintJS:** FingerprintJS was developed in 2012 and provides an open source fingerprinting library (the latest release is version 2). The motivation of this was mainly influenced by the Panopticlick project as well as PetPortal. Additionally, FingerprintJS has increased the use of various attributes (e.g., canvas fingerprinting, Timezone, CPU class, etc.) for improving the efficiency of the fingerprinting detection along with extending the ability to deploy with the most modern web browsers in China, e.g. QQ, Baidu and other.

- **Fybridjs:** Fybridjs[14] is an open-source fingerprinting tool. This method relies on different attributes of the web browser, without using any client-side identifier. This method uses JavaScript objects, canvas, plugin, and list of fonts attribute and then combine them into a fingerprint ID.

[13] https://panopticlick.eff.org/
[14]https://github.com/ammosh/fybridjs

Table 2-2 comparison of attributes used by fingerprinters

	Fingerprinters					
Attribute	***Pan***	***BC***	***IO***	***TM***	***Add***	***FPjs***
List of plugins	✓	✓	✓	✓	✓	✓
List of fonts	✓	✓	✓	✓	✓	✓
User-agent	✓	✓	✓	✓	✓	✓
HTTP Header Accept	✓					
HTTP Header Accept-Charset						
HTTP Header Accept-Encoding	✓					
HTTP Header Accept-Language	✓					
Screen Resolution	✓	✓	✓	✓	✓	✓
Timezone	✓	✓	✓	✓	✓	✓
Browser language	✓	✓	✓	✓	✓	✓
Operating system	✓	✓	✓	✓	✓	✓
DOM Storage	✓	✓	✓	✓	✓	✓
IE userdata	✓	✓				
Java Enabled	✓				✓	✓
DNT User Choice	✓				✓	✓
Cookies Enabled	✓		✓			
JS detect: Flash Enabled	✓	✓	✓	✓	✓	✓
ActiveX + CLSIDs	✓	✓	✓	✓	✓	✓
Date & Time		✓	✓	✓	✓	
CPU		✓	✓		✓	✓
System / User Language	✓	✓	✓		✓	✓
openDatabase			✓		✓	✓
Canvas Fingerprinting	✓				✓	✓
Mime-type Enumeration				✓		
HTTP Proxy Detection			✓	✓		

	Fingerprinters					
Attribute	***Pan***	***BC***	***IO***	***TM***	***Add***	***FPjs***
IndexedDB					✓	✓
Math Constants		✓			✓	
Windows registry		✓	✓			
TCP/IP Parameters		✓	✓			
Google Gears Detection		✓				
Flash Manufacturer				✓		
MSIE Security Policy		✓				
AJAX Implementation		✓				
MSIE Product key			✓			
Device Enumeration		✓	✓			
Device Identifiers			✓			
IP Address		✓				
HTML Body Behavior						✓
Battery					✓	
WEBGLRenderingContext					✓	

Fingerprinters	**cited from**
Pan	Panopticlick[15]
BC	BlueCava[16]
IO	Iovation[17]
TM	ThreatMetrix[18]
Add	AddThis[19]
FPjs	FingerPrintJS[20]

To consider Table 2-2, many fingerprinters use different attributes to fingerprint the user computer. The high entropy attributes (e.g. User-agent, list of plugins, list of fonts)

[15] https://panopticlick.eff.org/
[16] http://bluecava.com/
[17] https://www.iovation.com/
[18] https://www.threatmetrix.com/
[19] http://www.addthis.com
[20] https://github.com/Valve/fingerprintjs2

that are a significant factor for creating fingerprinting ID are used by six fingerprinters. As the low entropy attributes are somewhat diverse, this result shows the difficulty to guess which attributes are used for fingerprinting. Notably, more than half of attributes used by fingerprinters mainly use JavaScript to access the user data.

2.8 Summary of the chapter

The fundamentals of browser fingerprinting were described in Chapter 2. It began with the overview of current tracking technologies used for tracking a user. The author attempts to explain that now there is a new technique (the browser fingerprinting technique) used for tracking user without user awareness. Then, the definition of browser fingerprinting technique had been explained along with characteristic of browser fingerprinting technique, stability and diversity. Stability of fingerprint depends on a change of attributes used for fingerprinting, while the diversity of fingerprinting depends on the entropy attribute value.

The concept of identifying information is crucial importance to identify someone. When this concept applies to the browser fingerprinting technique by using actual attributes and features of the user's computer, it assists to track a user without leaving any files on the user computer. With the diversity of user's attributes, it assists the browser fingerprinting technique to identify a user uniquely through a use of web browser.

Many browser fingerprinting methods had been classified into categories and explained which categories are likely to be used in practice. Then, the overall process of data flows of browser fingerprinting technique had been illustrated in order to assist understanding of how the browser fingerprinting technique works. Finally, many attributes used by six popular fingerprinters had been illustrated. It can be clearly seen that over half of used attributes mainly use JavaScript to access the user data. Most fingerprinters select high entropy attributes to create their fingerprinting ID while the low entropy attributes chosen by tracking companies have made it difficult for fingerprint countermeasures to guess which fingerprint attributes should be handled.

(This page intentionally left blank)

Chapter 3 Review and Discussion of Existing Countermeasures

In recent years, developments in the field of fingerprinting prevention have been proposed by several researchers. Questions have been raised whether these proposed countermeasures are capable of inhibiting fingerprinting tracking or not. For this reason, this chapter consists of four main sections. In Section 3.1, it explains almost all possibilities of preventing the fingerprint tracking along with advantages and disadvantages of each countermeasure. Then, discussions about results of preventing fingerprint tracking by fingerprint countermeasures contribute to the problem of the information paradox clarified in Section 3.2. Afterwards, almost all countermeasures will be discussed with emphasis on which one best to use to prevent the fingerprint tracking in Section 3.3. In the final section, the future countermeasure that should be used will be discussed.

3.1 Discussion of possible countermeasures

The progress in the technique of web browser fingerprinting can threaten user privacy. Over a decade, the published research has suggested many ways of eliminating the ability of the web browser fingerprinting technique. Despite some countermeasure techniques going unused, some techniques are widely deployed. This section has collected relevant information, classified almost all fingerprint countermeasures and explained their different levels of success.

Browser fingerprinting technique usually consists of two key stages. The first stage is obtaining user data through various channels (e.g. JavaScript and plugins). The second stage is to combine obtained attribute values into one string and then calculate the tracker (the fingerprinting ID). The fingerprint countermeasure must select which stage should be addressed to defeat the fingerprinting algorithm. Based on the literature review, it can be classified almost all countermeasures into four types, namely blocking access to the user data, creating an identical identity, degrading uniqueness of fingerprinting ID and using the

browser extension. The aim of this chapter is to examine advantages and disadvantages of each countermeasure.

3.1.1 Blocking access to the user data

Blocking access to the user data is a straightforward method. FaizKhademi et al. (2015) attempt to explain that this method is to disable technologies employed for fingerprinting or to return an empty value of user's attributes to the fingerprinting server. When the fingerprinting algorithm is incapable of obtaining any value from a user computer, it cannot initiate the creation of fingerprinting ID. Once the fingerprinting ID is not sent back to the fingerprinting server, the fingerprinting server cannot actually identify the user computer as required. Several research studies have proposed the blocking technique as follows.

3.1.1.1 Disabled JavaScript

Eckersley (2010) and Boda et al. (2012a) who did research in the field of how to improve the efficiency of browser fingerprinting technique had strongly advised that an effective way to inhibit the fingerprinting technique is to disable JavaScript. JavaScript is the primary mechanism used for fingerprinting to access many of attributes within the user computer. If the fingerprinting technique is unable to access any attributes, a fingerprinting ID will not be created resulting from absence of attribute value. According to Fifield and Egelman (2015) who developed a web browser fingerprinting technique based on dimensions of font glyphs, they advised that the best way of defence against fingerprinting is to disable JavaScript, according to Eckersley (2010) and Boda et al. (2012a).

Unfortunately, most websites (e.g. Facebook, Google, etc.) now make extensive use of JavaScript. Disabling JavaScript causes directly demonstrable impact to these websites. In addition, much published research has pointed out problems with the user browsing experience caused by disabled JavaScript (Roesner et al., 2012, Chairunnanda et al., 2011). This method may prevent the browser fingerprinting technique but it is practically unfit for preventing the fingerprint tracking

3.1.1.2 Do not Track

"Do Not Track"[21] is a W3C standard also supported by the EU Cookie Directive (Tschofenig and van Eijk, 2011). The concept of Do Not Track utilises an HTTP header field with three values: '1' a user does not wish to be tracked, '0' a user allows tracking or 'null' a user did not set a preference. This value will be sent with a request message by the web browser to notify the web server to inhibit tracking. If the web server follows this standard, the web browser will not be tracked by the web server. This feature is implemented in all modern web browsers (e.g. Chrome, Internet Explorer or Firefox). However, Acar et al. (2013) established that this method is unable to prevent fingerprinting tracking because many fingerprinting servers simply ignore the signal of Do not Track. There is currently no enforcement (Balebako et al., 2012) and this solution can only be effective if policymakers take a decision to require all websites to comply with the standard and an effective enforcement regime is imposed (Broenink, 2012b). Even though the General Data Protection Regulation (EU) 2016/679 (GDPR)[22] has come into force, the enforcements is still a challenge to force servers to respect the signal of Do Not Track.

3.1.1.3 Allowing content before execution

By this approach, any scripts executed on the web browser will be allowed by a user. The user can block executable web content such as Java, JavaScript, Flash and other plugins. When a user browses any web page, the active content would typically be blocked by default. The users then decide which scripts should be executed on their computers (Malandrino and Scarano, 2013). For example, NoScript is a browser extension designed to block all executable web content automatically. The user must then take a decision as to which scripts should be executed by allowing explicit permission on an impermanent or permanent basis.

However, the running of the web page is unlikely to operate smoothly while rendering a web page because scripts are blocked by default. In addition, recognising that a web page

[21] http://donottrack.us/
[22] https://gdpr-info.eu/

does not contain any threat is significantly more complicated than in the past. Some websites load only from their own domain, but others load from third-party servers for various reasons (e.g. to display advertisements or to track the user). This complicated operation confuses users without the necessary experience of website technologies. In addition, Mowery et al. (2011b) established that some fingerprinting web servers could potentially exploit the utilisation of a whitelist by using the websites allowed in the whitelist as one attribute to create a fingerprinting ID.

3.1.2 Creating an identical identity

Normally, almost all users' computers have an identity difference (Torres et al., 2015). These computers are configured by the user with difference in use of operating system name or version, screen size, plugins and so on. Supposing all computers had the same identity, the fingerprinting algorithm could not distinguish which computers belonged to any user (Mu and Hu, 2012). With this concept, this technique would modify attributes on the user computer to an identical identity. In practice, this approach does not resist any attempts to access the user data by the fingerprinting server. The attribute values returned back to the fingerprinting server will be faked, and thus the fingerprint ID created by the fingerprinting algorithm will be a duplicated fingerprint ID that will no longer identify the user.

3.1.2.1 Agreement to use common APIs.

One mechanism that assists the fingerprinting web server to identify the user computer is the use of a number of different APIs and engines for each web browser. For example, there are many vendors involved in the improvement of JavaScript engines on a web browser, and all modern web browsers are equipped with different engines which result in different responses to the web server. The web server can detect the rate of response to the web server and then use the different requests and replies as an additional attribute to fingerprint the user computer (Mowery et al., 2011a). One way to tackle this problem is to force every web browser vendor to use the same set of APIs (Upathilake et al., 2015). If all web browsers use the same standard APIs and engine, the fingerprinting technique cannot distinguish any difference between the request and response messages which eliminates this source of potential fingerprinting.

However, this approach appears to be too difficult to implement. Many vendors are unwilling to follow this suggestion because they are concerned about such potential difficulties as the possible lower performance of their web browser (Nikiforakis et al., 2013).

3.1.2.2 Sharing same fingerprinting ID with others

Nikiforakis et al. (2014a) established that smartphones frequently shared the same fingerprint ID because they have few attributes that can be accessed by the browser fingerprinting technique. For this reason, sharing the same fingerprinting ID would become a problem for the browser fingerprinting technique in distinguishing a user.

To consider the unique identity of the user computer created by the browser fingerprinting technique, it is one of the most significant factors for identifying a user computer. Assuming if web browser A's identity is not different from web browser B's identity, the fingerprinting server is incapable of distinguishing the difference between web browser A and web browser B. With this reason, the concept of degradation of the uniqueness of fingerprinting ID is presented. This concept attempts to degrade the unique identity of the web browser by modifying and restricting attributes on the web browser until the web browser's modified identity is homogenous. If all web browsers have the same identity, the fingerprinting technique cannot distinguish a user.

With regard to this concept, Tor is a great example. The concept of Tor is to modify, disable and restrict many attributes (e.g. a list of fonts, a list of plugins, User-agent, some browser APIs, etc.) on the web browser. The results of attribute modification and restriction cause all Tor users to have the same identity[23]. The fingerprinting techniques cannot identify any Tor browsers because they will all appear to be the same browser.

Unfortunately, the speed of Tor is significantly slower than the speed of regular web browsers due to the nature of its connections. In addition, Tor hides the user's location and randomly picks the server's location which can cause problems for the login system

[23] https://www.torproject.org/projects/torbrowser/design/

(Lenhard et al., 2009), and Tor users are treated as second class on the Web, frequently being denied services by many websites (Khattak et al., 2016). In addition, if the uniqueness of Tor is considerably decreased, users can be detected as a group (Broenink, 2012a). Moreover, this concept is developed further by Fiore et al. (2014). They attempt to construct a fake profile with Google Chrome by modifying many attributes such as HTTP header and Navigator object. The modified Google Chrome has the same identity, a similar concept to the Tor browser, which they claim can deter fingerprinting tracking. Unfortunately, they did not show any results of experiments nor the research methodology of how their countermeasure can prevent fingerprinting tracking.

3.1.3 Degrading uniqueness of fingerprinting ID

The uniqueness of the fingerprinting ID frequently depends on attributes detected on the user computer. In order to undermine the uniqueness of fingerprinting ID, a user computer needs to limit access to user attributes, especially high entropy attributes. The result is to undermine the uniqueness of fingerprinting ID.

3.1.3.1 Decreasing use of plugins

The list of plugins in a web browser provides a relatively-high-entropy attribute, around 15.4 bits of entropy as measured by Eckersley (2010). This list is significant not only because it makes an important contribution to the attributes available to a fingerprinting web server, but it also reveals underlying user information through plugin APIs, (e.g. a list of fonts, operating system, screen resolutions, etc.). The Tor browser decreases the use of plugins by allowing only selected plugins, hence reducing leakage of user information. Another example is a smartphone browser which typically allows only a few plugins. This implicitly reduces the entropy value, and it is harder for a fingerprinting website to create a unique tracking ID (Nikiforakis et al., 2014a). Boda et al. (2012b) noted that the increasing number of plugins had improved the robustness of fingerprinting algorithms, hence decreasing the number of plugins should help to prevent fingerprint tracking. In addition, Natalia[24] explained that the inability to install plugins on a mobile phone makes

[24] http://akademie.dw.de/digitalsafety/your-browsers-fingerprints-and-how-to-reduce-them/

fingerprinting harder as well as Nikiforakis et al. (2013) who concluded that regardless of the complexity of plugins there was an inability to control user data.

Although degrading the stability of fingerprinting by decreasing the number of plugins appears to be an attractive method, the consequence of disabled plugins can produce adverse effects on the browsing experience. Any plugin deficiency can result in the problems affecting the smooth and proper running of the web page. For example, installing a PDF plugin will assist the user to view PDF documents conveniently. If the browser did not provide a built-in PDF facility, users would be unable to view PDF files easily.

3.1.3.2 Decreasing use of fonts

Each user computer has different fonts used for different purposes by the user. The list of fonts is classified as a high entropy attribute, 13.9 bits of entropy as measured by Eckersley (2010). Some countermeasures attempt to limit access to the list of fonts with the purpose of degrading the uniqueness of fingerprinting ID. For example, Tor allows only some fonts capable of running on a Tor Browser[25], not all fonts. The main purpose of this is to undermine the potential of fingerprinting technique to have sufficient attributes to create a unique fingerprinting ID. In addition, font is also classified as an independent attribute. For this reason, any alteration of fonts does not affect other attributes on the user computer.

3.1.3.3 Using multiple browsers

This approach makes use of different browsers for different activities when surfing the Web. User profiles will be more difficult to create if the user makes use of more than two browsers. For example, suppose one browser is used only for Facebook, Google+ and e-commerce, and another one is used for general browsing. This concept assumes that the user profile is more difficult to build up if users can change their web browser. Despite the seemingly-straightforward concept, it provides only a partial solution because a fingerprinting web server can select other attributes unrelated to a web browser (e.g.

[25] https://www.torproject.org/projects/torbrowser/design/

operating system, underlying hardware, screen resolution or IP Address) to fingerprint the user computer rather than relying on browser attributes (Boda et al., 2012a).

3.1.4 Using browser extension

A web browser can be developed with a new functionality by using a browser extension, such as altering appearance or content of websites. A developer can create a new extension with the use of core web development technologies such as HTML, JavaScript and CSS. The created browser extension can be uploaded to online stores which permit normal users to find created browser extensions on the Internet.

Modern web browsers (e.g. Firefox and Chrome) have allowed developers to extend browser functionality for various purposes such as enhancing security, inhibiting advertisements, and adding new features (Shahriar et al., 2014). Browser extensions can be used to enhance user privacy by monitoring and intercepting activities (i.e. it can inject the Logger component into the Document Object Model (DOM) before running a web page) on the web browser while the web page has begun loading. With this ability, the browser extension is a way to inhibit the fingerprint tracking.

The first part of this section justifies the traditional process of developing the browser extension into a fingerprint countermeasure. The second part goes into detail about selecting attributes and methods for changing attributes.

3.1.4.1 The overall process of developing existing countermeasures

From the literature review, almost all fingerprint countermeasures based on the use of browser extensions have been developed based on only the two main steps as shown in Figure 3-1.

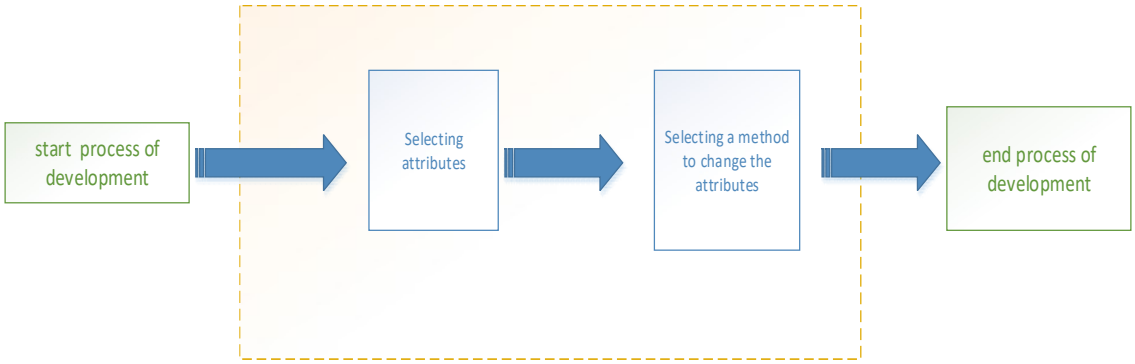


Figure 3-1 the two main parts of developing the existing countermeasures

Step 1: Selecting attributes

In a computer, attribute is used to refer to a specific characteristic, usually comprising a name and value. Similarly, JavaScript provides numerous objects. Each object consists of a name and value. The value of the object can be added, deleted or changed as required by a developer.

Typically, there are many attributes on the user computer used for various reasons. For example, the plugin attribute is used to improve the new functionality on the web browser for customizing the display of the web page. Among the large number of attributes, some attributes have both completely different, while other are slightly different in term of characteristic. The degree of uncertainty of information in the object is one factor which has considerably influenced the diversity of fingerprinting ID. As mentioned in Chapter two, the entropy is a way to measure the degree of uncertainty of information content in units of bits. The entropy value of an attribute is a significant source to create the uniqueness of fingerprinting ID. There is no restriction on how many attributes should be used for inhibiting the fingerprint tracking. Thus, choosing different attributes of each fingerprint countermeasure can have a substantial impact on the running of the web browser. This section classifies countermeasures into three types as follows.

1. **Selecting only high entropy attributes**

According to published research, the list of fonts, a list of plugins and User-agent are classified as high entropy attributes (Eckersley, 2010, Boda et al., 2012a, FaizKhademi et al., 2015). Thus, the concept of selecting only the high entropy attributes would imply that it is not necessary to change other attributes, especially low entropy attributes. For example, if they change only plugins (which according to the Panopticlick project is 15.4 bits), they could compromise the process of fingerprinting to create the unique fingerprinting ID.

2. Ignoring actual entropy values

With regard to ignoring actual entropy values, this method would change suspicious attributes regardless of their entropy. The concept behind this method is that suspicious attributes must be changed immediately. Ignoring entropy attribute value appears to be a straightforward method but causes significant side effects to the web browser.

3. Selecting appropriate attributes

The last method is selecting an appropriate attribute. The concept of this method is to attempt to fix the problem of previous methods. Selecting only high entropy attributes has a weakness that remaining attributes may be used for fingerprinting. In order to fix this problem, this method would select both suspicious low entropy attributes and high entropy attributes. The main idea is that suspicious attributes related to fingerprinting should be changed. This method can fix the problem of remaining attributes to be used for fingerprinting and the information paradox of changed attributes on the web browser. However, the main problem is that there are many suspicious attributes on the user's computer that should be changed. This method cannot use all suspicious attributes for change. Thus, the process of selecting suitable attributes depends on the idea of each countermeasure which attributes should be used for change.

Step 2: Selecting a method to change attribute

After definitely selecting attributes, many fingerprint countermeasures would select which techniques should be used for changing selected attributes. In the past decades, a number of researchers have sought to propose several techniques that can be classified

into three types: randomization, blocking and spoofing. Each technique has advantages and disadvantages as described below.

1. Randomization

The randomization technique is a technique for breaking linkability. The term linkability is a relatively new name for the field of fingerprinting technique, commonly referred to as – "*the capability to connect the same fingerprinting ID across multiple visits" (Nikiforakis et al., 2014b)*. This technique realises that the real culprit is not the uniqueness of fingerprinting ID, but it is linkability – the ability to link the same user over multiple visits with the same fingerprinting ID. If linkability is neutralized, the fingerprinting technique cannot use the obtained fingerprint ID to track the user anymore.

Another meaning of randomization is a process of generating something at random. Because fingerprinting ID usually derives from user's attributes, alteration of any attribute values on the user's computer can directly affect the fingerprinting ID, usually sent back to the fingerprinting server. With changed fingerprinting ID, it can obfuscate the process of identifying a user on a fingerprinting server. As well as Besson et al. (2014) who were interested in using randomisation technique. They proposed privacy enforcement mechanism based on randomisation against fingerprinting but their method was a just only theory, no concrete results.

In a technical context, the browser extension will be developed to intercept running of a web page by seeking required attributes, and then changes those attribute values every time before displaying a web page (FaizKhademi et al., 2015). As a result, the fingerprinting ID will be changed at every visit to a web server.

In consideration of advantages and disadvantages of the randomization technique, this technique is seemingly straightforward. It just chooses attributes that it is required to change and then changes an attribute value before running a web page. The result of the change could generate the new fingerprinting ID (the fake fingerprinting ID) which would be sent back to the fingerprinting server. However, the poor randomization policy may affect the users' experience because some changed attributes may produce negative side

effects to the web browser. For instance, if the web browser pretends to change an attribute of User-agent from Windows to iPhone. The web page may display incorrectly because a web server understands that the user employs iPhone platform on its device. As noted by Acar (2014), a fingerprint countermeasure should be more careful about changing an attribute value because some attributes may be associated with other attributes. The arbitrary randomness may increase problems rather than solving the problem of fingerprint tracking

2. Blocking Attributes

The blocking approach is to disable attributes used for fingerprinting (e.g. JavaScript object, HTML5 canvas, etc.) or to assign an empty value to attributes (Weldemariam, 2015). A consequence of the absence of attribute value affects the fingerprinting algorithm to be unable to bring any values to generate the fingerprinting ID for tracking the user. This approach would intercept loading of a web page. Then, it would seek required attributes by changing the value or assigning the empty value to the attribute and then allows the web browser to run as usual. The blocking technique is a seemingly effective way to resist the fingerprint tracking.

However, the absence of attribute value on the user's computer could compromise running of a web browser and lead to problems of the user experience. According to Olejnik et al. (2012), the blocking technique causes direct impacts to the user experience and it is difficult to avoid functionality loss.

3. Spoofing attributes

The spoofing approach is to modify attributes once. This approach will not change attributes every time a web browser requests to a web server. In the concept of spoofing technique, a user being fingerprinted within one website is not as dangerous as bringing the fingerprinted user's identity to use on other websites for tracking a user. The heart of this concept is to allow a website to create the user's identity within one website, but not allowing it to bring the fingerprinted user's identity to use with other websites. With this concept, attributes are not changed if the user does not change URLs of a website, but attributes are changed when a website's URLs change. The concept of this technique is

similar to the randomness technique — *considering that it is not uniqueness of fingerprinting ID that harms user privacy, but linkability*.

However, in order to recognize which websites have ever been visited by a user, the whitelist will be created in order to keep a number of websites that have been visited. This reason causes a fingerprinter to be able to use a whitelist to fingerprint a user as a normal attribute (Mowery et al., 2011b).

3.1.4.2 Existing fingerprint countermeasures

This section attempts to explain the concept of each fingerprint countermeasure. With different perceptions, each fingerprint countermeasure has different ideas to choose attributes and different methods of changing attributes.

1) User-agent spoofing

User-agent normally is a string used for telling a website about the basic information of the operating system and web browser. Most websites would personalize their web pages consistent with the information obtained through User-agent. Typically, each web browser has its own User-agent. Thus, each user would possess different User-agent, depending on the name and version of a web browser. When a browser connects to any websites, it would also send a User-agent to HTTP header to a website. A User-agent contains many attributes within one string of the User-agent (e.g. browser name, version of web browser, operating system, CPU, etc.). The contents of User-agent filed vary from a web browser to web browser. With different information inside of the User-agent, it can be used for fingerprinting.

Yen et al. (2012b) suggest that changing the User-agent is an approach to avoid fingerprint tracking as it makes fingerprinting less unique. A number of extensions have been recently made available by developers (e.g., UserAgent Switcher, UserAgent RG, UAControl, or UserAgentUpdater) for various purposes. For example, spoofing the User-agent not only assists the user to view a website properly as originally designed, but it also helps a webmaster to test the web resources with other User-agents. One approach is to alter User-agent information to inhibit fingerprint tracking.

However, Nikiforakis et al. (2013) noted that using User-agent spoofing often results in unwanted side-effects and increases the risk of being fingerprinted by somebody. User-agent is an attribute related to other attributes. Changing User-agent can lead to an impossible configuration (e.g. mismatching of the User-agent information with the JavaScript layer) which ironically could result in an easier identification of the web browser (Eckersley, 2010, Nikiforakis et al., 2014c).

2) Privaricator

Privaricator is a countermeasure designed for preventing the fingerprinting technique by employing the principle of random policies (Nikiforakis et al., 2014b). It assumes that the user will visit websites for the first time. Visiting a website with the same browser the information returned will be altered in a random fashion every time so that each visit appears to be from a new user. This technique focuses on the list of plugins and list of fonts (selecting only the high entropy attribute) by using the method of randomness to hide plugins and fonts and also adds some random noise to the offset properties.

However, this countermeasure neglects to change the User-agent by claiming that changing only list of plugins, fonts, and adding noise to the offset properties are sufficient for fingerprinting prevention. The controversy about ignoring low entropy raised concerns that the remaining attributes may provide a soft target for fingerprinting websites. In addition, this countermeasure failed to engage with the canvas fingerprinting. Nikiforakis et al. (2014b) cited the experiment of Acar et al. (2014) that no fingerprinting companies used canvas fingerprinting.

3) RubberGlove

RubberGlove[26] is a Chrome extension which can be installed from the Chrome web store. It is designed to find JavaScript objects (navigator and screen) within a web page and then replaces them with null values. This countermeasure attempts to block the access of JavaScript objects which cannot then be used to create a fingerprinting ID. This fingerprint

[26] https://chrome.google.com/webstore/detail/rubberglove/koabfojebhfdjnligkcihoeekimoekpg?hl=en

countermeasure is irrespective of entropy value. It engages all suspicious attributes that may be used for fingerprinting.

4) FP-Block

FP-Block is designed to use spoofing techniques for changing both high and low entropy attributes on the user computer (Torres et al., 2015). FP-Block uses the concept of web identities by changing attributes on the user computer if the user changes URLs of a website. FP-Block is the first countermeasure to establish its own model (FP-Block) to change attributes based on the consistency of attributes. The changed attributes by FP-Block model are expected to be close to the attributes of an unmodified web browser. This countermeasure is not only designed to prevent fingerprint tracking, but also provides a new way to mitigate the problem of a web browser standing out. However, the concept of FP-Block is based on the spoofing technique which is at high risk for making a web browser easier to fingerprint (Laperdrix et al., 2017).

5) Fireglove

Fireglove is a Firefox extension specifically designed to use blocking technique and randomization technique. Fireglove selects both high and low entropy attributes. It changes the User-agent and platform, returns a random value for the offsetWidth and offsetHeight parameters, disables both plugins and mimeType and restricts loading of fonts and offset value.

However, Acar et al. (2013) discovered that they could use another instruction (getBoundingclientRect) to disclose the value of offsetWidth and offsetHeight and the fix of a number of loaded fonts are not workable as suggested.

6) CanvasFingerprintBlock

CanvasFingerprintBlock is a Chrome extension which is specifically designed to block canvas fingerprinting by detecting canvas script with finding keywords. It detects .toDataURL() which is an instruction to write a canvas element. If it finds a suspicious keyword, it will send a message to notify the background page and display a deleted symbol on the Chrome browser.

7) FPGuard

FPGuard (FaizKhademi et al., 2015) implements a number of algorithms to analyse fingerprinting at runtime. FPGuard selects both high and low entropy attributes and is designed to prevent four types of fingerprinting technique. Changing attributes depends on the detection part of FPGuard which is designed to monitor the loading of suspicious attributes. If FPGuard can detect abnormal loading of suspicious attributes, it would change them with the randomization technique. For example, if it discovers object fingerprinting, it will randomize the attributes. Another example is if it detects fingerprinting with a list of plugins, it will generate virtual plugins in order to obfuscate the fingerprinting detection. Unfortunately, FPGuard is not made publicly available to download. The research is only a study of its concept because the results of its fingerprint prevention is so interesting to explore.

3.1.4.3 Comparison of existing countermeasures.

A number of existing countermeasures based on browser extension use different ideas and techniques to prevent the fingerprinting tracking. In order to understand the overall view, this section compares and contrasts existing countermeasures as follows.

1) Difference of detection and prevention part

Fingerprinting countermeasures based on browser extension typically consist of two main sections, detection and prevention, as shown in Table 3-1. To consider Table 3-1, it can be clearly seen that a fingerprint countermeasure does not necessarily use both prevention and detection. Some countermeasures (e.g. RubberGlove) may select only the prevention part which seems adequately efficient for resisting the fingerprint tracking.

Regarding the reason why some fingerprint countermeasures choose detection part to prevent the fingerprinting technique. Usually, there is no standard criterion for which fingerprinting attributes should be selected in fingerprinting detection. Some fingerprint countermeasures perceive that the accurate detection could lead to the increasing effectiveness of fingerprinting prevention and reducing side effects on the web browser. For example, Privaricator observes a number of loading of fonts and plugins as an indicator

of fingerprinting (Nikiforakis et al., 2014b). They perceived that knowing the movement of attributes could assist fingerprinting prevention with minimum impact to the web browser.

Table 3-1 the parts of each countermeasure

Countermeasure	Detection	Prevention
UserAgent Spoofing	×	✓
Fireglove	×	✓
Privaricator	✓	✓
FPGuard	✓	✓
FP-Block	✓	✓
CanvasFingerprintBlock	✓	✓
RubberGlove	×	✓

However, many fingerprint countermeasures operate at runtime. The results of detecting suspicious attributes are based on an automatic process of detecting suspicious attributes. This may identify the process of fingerprinting incorrectly. According to Acar (2014), many fingerprinters deliberately attempt to hide their fingerprinting code, and fingerprinter‘s code is extremely difficult to understand even Acar (2014) used a manual method to analyse. Many researchers had proposed a method to identify the fingerprint tracker. For example, FaizKhademi et al. (2015) proposed a method of suspiciousness level to guess the probability of fingerprinting. However, they honestly admitted that their algorithm cannot detect precisely.

In consideration of the prevention part, this part is responsible for changing the user’s attributes. The results of the detection part will be passed through to the fingerprinting prevention in order to change the user’s attributes in accordance with the detection part. Using a detection part is seemingly good for helping the prevention part to identify fingerprinting. In contrast, if the detection part identifies fingerprinting incorrectly, the prevention part will change suspicious attributes inconsistent with the reality.

2) Difference of selecting attributes and methods

Regarding Table 3-2, many fingerprint countermeasures select different methods and different entropy attributes to change user's attributes. This table attempts to explain how many attributes are used for detection and prevention. Only three countermeasures claim that they are careful about the side effects to the web browser.

Table 3-2 comparing the difference of each countermeasure

		Detection part	**Prevention part**		
Countermeasure	**Type of method**	**Attribute used for detection**	**Selecting high entropy**	**Selecting low entropy**	**Considering side-effects**
User-agent spoofing	Spoofing	-	✓ (1 attributes	-	-
Privaricator (Nikiforakis et al., 2014b)	randomization	✓ (5 attributes)	✓ (2 attributes)	✓ (3 attribute)	✓(by avoiding change of User-agent)
RubberGlove[27]	Blocking	-	✓ (3 attributes)	✓ (7 attributes)	-
FP-Block (Torres et al., 2015)	Spoofing and Blocking	✓ (19 attributes)	✓ (2 attributes)	✓ (17 attributes)	✓ (using concept of web identities
Fireglove (Boda, 2014)	Blocking and randomization	-	✓ (3 attributes)	✓ (5 attributes)	-
CanvasFingerprintBlock[28]	Blocking	✓(1 attributes)	-	✓(1 attributes)	-
FPGuard (Weldemariam, 2015)	Randomization	✓ (32 attributes)	✓ (3 attributes)	✓ (approximate 12 attributes)	✓(randomizing virtual plugins)
Don't FingerPrint Me[29] (2018)	Spoofing	-	✓ (1 attributes	-	✓(avoiding change many attributes)
BP Privacy Block All Font and Glyph Detection[30](2018)	Blocking	-	✓ (1 attributes	-	✓(avoiding change many attributes)

[27] https://chrome.google.com/webstore/detail/rubberglove/koabfojebhfdjnligkcihoeekimoekpg?hl=en
[28] https://chrome.google.com/webstore/detail/canvasfingerprintblock/ipmjngkmngdcdpmgmiebdmfbkcecdndc
[29] https://chrome.google.com/webstore/detail/dont-fingerprint-me/nhbedikkbkakbjipijipejfojanppbfg
[30] https://chrome.google.com/webstore/detail/bp-privacy-block-all-font/bfoidacjeobbnkpoemlfllfmmnbogcig?ref=s

Currently, there are few countermeasures that are released such as Don't FingerPrint Me[31] or BP Privacy Block All Font and Glyph Detection[32]. However, these countermeasures seem to use uninteresting methods to deal with the fingerprint tracking.

3.2 Information Paradox on a web browser

Eckersley (2010) was the first to introduce the problem of the information paradox (or abnormal combination). The data that was collected during his experiment showed fascinatingly strange information. His investigation pointed out that some web browsers possessed an abnormal combination which is absolutely impossible to be a real value of a web browser. For example, some web browsers showed information on HTTP header to be an iPhone, but JavaScript APIs showed a Flash plugin to be installed, which is not provided on an iPhone. He attempted to point out that if a user attempts to hide or change information, the user may make themselves more distinguishable from the crowd if the used technique is not shared by a sufficiently large set of users.

Later, the information paradox has come under additional study by Nikiforakis et al. (2013). They established that many spoofing extensions that are designed to conceal the true nature of a web browser have various problems. He had downloaded 11 popular extensions. His result showed in accordance with Eckersley (2010) that many web browsers that installed a spoofing extension are unable to hide their true nature. In addition, he had concluded three main problems of information paradox as follows.

- Incomplete coverage of navigator object: many browser extensions frequently intend to spoof a JavaScript object, but they turn out to leave some traces that someone can observe. For example, some User-agent spoofing browser extensions

[31] [31] https://chrome.google.com/webstore/detail/dont-fingerprint-me/nhbedikkbkakbjipijipejfojanppbfg
[32] https://chrome.google.com/webstore/detail/bp-privacy-block-all-font/bfoidacjeobbnkpoemlfllfmmnbogcig?ref=s

attempt to fake User-agent, but they do not fake AppName and Appversion which are contained in the JavaScript navigator.

- Inconsistencies between related attributes: For example, if a web browser is altered a screen object from the personal computer into the mobile phone. Fingerprinters may capture the abnormal condition from other attributes such as window.screen. The unusual combination makes them find other ways to fingerprint a user.

- The mismatch between HTTP header: Normally, the HTTP header will be sent automatically to the client. Therefore, if some attributes are modified in application layer, the information on the HTTP header will show that it is incompatible with the application layer. For example, if a user seeks to spoof the operating system in the application layer, they need to recognise whether this attribute is also contained in the HTTP header.

There are only fingerprint countermeasures that attempts to control side effects of information paradox. In spite of no concrete result of side effects from problems of information paradox by Eckersley (2010) and Nikiforakis et al. (2013), it needs to accept that the information paradox is one of the problems occurring in changing attributes by a fingerprint countermeasure. Neglecting this problem makes a user easier to be fingerprinted. Andalibi et al. (2017) attempt to study this problem of information paradox by using Markov model. Unfortunately, they did not provide any concert results.

3.3 Possible way to prevent the browser fingerprinting technique

Creating a browser extension appears to be an attractive avenue to answer the research question. Other ways seem to offer simple techniques, not a solution. Users cannot confidently rely on these techniques if they must browse the web, and some techniques seem impossible to implement in practice.

Disabling JavaScript is quite hard to achieve in practice (Roesner et al., 2012, Chairunnanda et al., 2011) because almost all websites use JavaScript to run. Even though users can prevent a fingerprinting algorithm, users will face the inevitable consequences of disabled JavaScript (e.g. functionality loss).

Regarding the agreement to use common APIs, it may take a long time to achieve because each modern web browser requires development in their own way to achieve their own success (Nikiforakis et al., 2013).

Decreasing use of plugins did not solve the entire problem of fingerprint tracking. It just decreases the uniqueness of fingerprinting (Doty, 2016). This method only handles attributes that contribute to the fingerprintability. However, this method can be applied to other countermeasures in the future.

Using multiple browsers still leaves vulnerability to attack by fingerprinting techniques. It may use other attributes unconnected with browser attributes to create the unique tracker (e.g. screen or platform) (Boda et al., 2012b).

Sharing a fingerprint with others seems a good idea but no research currently shows a concrete result that this is a good method to prevent the fingerprinting technique. In addition, the effectiveness of fingerprint prevention depends on the number of users sharing the same fingerprint. If the number of users using this method decreased, the users have a risk of being fingerprinted. According to Broenink (2012a), this technique has a chance to identify users as a group.

All of these problems of current methods should be replaced with a browser extension. There are many reasons to support this idea.

1. **Unnecessary to disable JavaScript**

   Loading of web pages can be intercepted and detected by using the browser extension[33]. The web page modified can run JavaScript the same as an unmodified web page. This is a good solution to keep the browsing experience and still prevent the fingerprint tracking while users are browsing.

2. **Solving fingerprinting problem with own method**

   As for expecting someone, or the modern web browsers, to negotiate and agree a common solution in order to use the common APIs, it seems far-fetched to achieve

[33] https://developer.mozilla.org/en-US/Add-ons/Overlay_Extensions/XUL_School/Intercepting_Page_Loads

(Nikiforakis et al., 2013). The new countermeasure should find a possible way relying on one's own ability rather than expecting someone else to tackle the problem of fingerprint tracking.

3. **Possibility to tackle entire problem of fingerprinting**

   Many countermeasures have been made based on using a browser extension (Nikiforakis et al., 2014b, FaizKhademi et al., 2015) to deal with the problems of the fingerprinting technique. This solution seems to tackle the entire problem of the fingerprinting technique rather than reducing plugins or using multiple web browsers.

4. **Improving the browsing experience**

   Installing a browser extension does not affect the services of typical websites. The user will be treated as a normal user. It contrasts with the Tor browser where users will be treated as second class on the Internet[34].

3.4 Summary of the chapter

This chapter has reviewed and classified almost all fingerprint countermeasures proposed to the public (and some are widely available on the Internet). Despite having proposed countermeasures, their ideas seem too far-fetched to implement in the real world. One solution that seems feasible is to use a browser extension to inhibit fingerprinting at runtime. However, fingerprint countermeasures based on a browser extension have different concepts to handle the fingerprint tracking. With different concepts to handle, each countermeasure could possibly produce different side effects to the web browser. These countermeasures mainly use three basic techniques (blocking, spoofing, and randomization technique). These techniques play a crucial role in changing the fingerprint so as to break the linkability while browsing websites. Unfortunately, many existing countermeasures have focused only on the prevention technique without paying attention to the information paradox and browsing experience.

[34] https://browserprint.info/blog/fingerprintingDefence#ref1

Chapter 4 Research design

In the previous chapters, the aim was to provide the background knowledge related to this research. Prior to the chapter that shows the results of the study, it is vital to explain the overall plan of this research. The overall plan, or research design, is needed by all good research to provide valid research findings (Kumar and Phrommathed, 2005). Thus, the overview of research design will be described in this chapter in order to increase understanding of the research. Section 4.2 – 4.4 gives an overall plan that comprises the main tasks to be conducted. The steps of the problem identification will be described briefly in Section 4.2 and then it will be explained why the research has to conduct preliminary experiments before proposing a new fingerprint countermeasure in Section 4.3. Finally, all processes of the evaluation of the proposed countermeasure will be explained further.

4.1 Overview of the research design

The basic concept of privacy is broadly relevant in many fields and different disciplines such as law, management and computer science (Piao et al., 2016). This research can be classified as a computer science. The research work conducted in this thesis is based on the design science research process in computer science. A normal sequence of activities can be classified into three main phases: problem identification, solution design and evaluation (Offermann et al., 2009).

In accordance with Peffers et al. (2007), a design science research process consists of six activities: problem identification, the definition of objectives of a solution, design and development, demonstration, evaluation and communication. The overall details are summarised in Table 4-1.

A design science research process of this thesis is shown in Figure 4-1. In the first phase, the problems of an Internet user being tracked by the browser fingerprinting technique would be considered as a problem of user privacy. Later, the research would propose a way to address the fingerprint tracking.

The second phase is designed to address the problem in the first phase. The research would propose a new concept to inhibit the fingerprint tracking in this phase. The main aim of this phase is to prevent the fingerprint tracking and have a minimum impact on the user browsing experience.

In the final phase, the proposed fingerprint countermeasure would validate the effectiveness of fingerprinting prevention, user experience, information paradox and the performance overhead of the proposed fingerprint countermeasure. The results of this phase require that the proposed fingerprint countermeasure can address the problems of existing countermeasure.

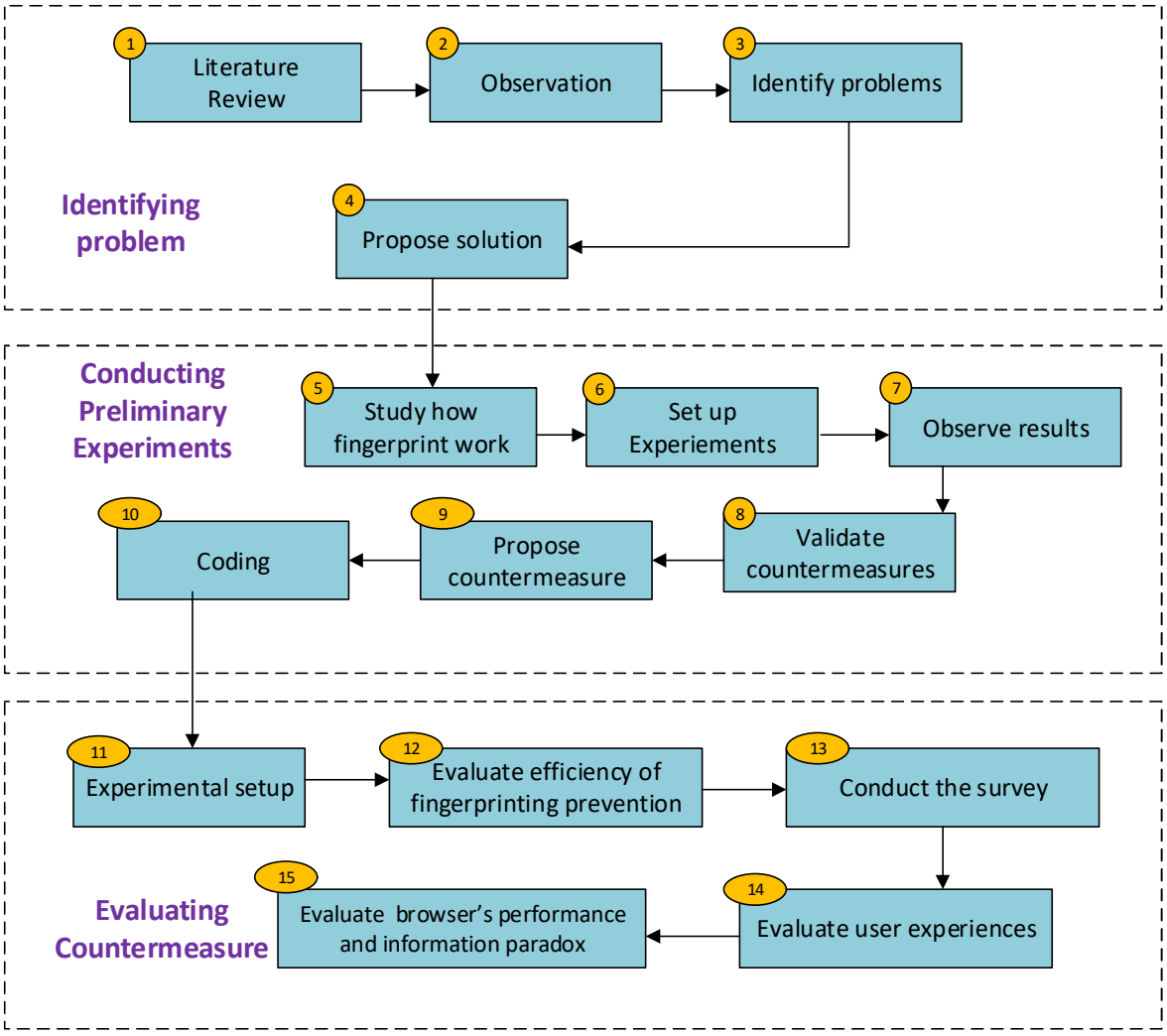


Figure 4-1 the overview of research design

Table 4-1 overview of design science research activities demonstrated in Peffers et al. (2007)

Phase	Design research activities	Description	Output
Problem identification	Problem identification and motivation	Finding the research problems	Research question and a countermeasure
	Definition of the objectives of a solution	Determining which countermeasure should be improved further	
Solution design	Design and development	Studying how the fingerprinting works and studying strengths and weaknesses of current countermeasures.	The improved fingerprint countermeasure
Evaluation	Demonstration	Use the improved fingerprint countermeasure to validate by comparing to current countermeasures	
	Evaluation	Determining the improved fingerprint countermeasure by comparing results with others	Results
	Communication	Report results, limitation of new knowledge	PhD Thesis

4.2 Problem identification

Understanding of the current problems of user privacy and development of trackers for tracking a user is essential for the starting point in this thesis. The research began from three literature review papers by Eckersley (2010), Boda et al. (2012a) and Acar et al. (2013). The first review paper by Eckersley (2010) had shown the possibility for the user to be tracked by the fingerprinting technique. The second paper by Boda et al. (2012a) showed selecting of some attributes of the user computer that can enhance the capability of the fingerprinting technique. The final paper by Acar et al. (2013) showed that a number of websites have recently adapted the fingerprinting technique for tracking users in the wild. These three papers indicate that the great strength of the browser fingerprinting technique is to track users without user's awareness and permission, and users have difficulty to avoid tracking. Currently, many commercial companies have paid great attention to the browser fingerprinting technique for various purposes. This technique is probably going to be applied by their businesses for tracking users in the future.

A further literature search found two research papers by Nikiforakis et al. (2014b) FaizKhademi et al. (2015) that it is probably possible to inhibit the fingerprinting tracking by changing some attributes of user computer before displaying a web page. However, it seems likely strange because two research papers described only advantages of their countermeasures. They did not present their limitations while using their countermeasures. According to Laperdrix et al. (2015) and Torres et al. (2015), they state that when changing attributes, one needs to be careful about the combination of the changed attributes. If the relationship of attributes of a web browser is not consistent, it may make problems with the web browser rather than preventing the fingerprint tracking.

The literature search attempted to find a new way to inhibit the fingerprinting technique. Many solutions seem far-fetched to implement in practice. Furthermore, much published research pointed out that many existing countermeasures still have problems with the user browsing experience and information paradox. However, using browser extension still seemed to be the best way to address the problem of fingerprint tracking.

4.3 Solution Design

This is to ensure that all of the information of problem identification phase is correct. This phase will be divided into three main sections: studying of the workflow of fingerprinting technique, studying of fingerprint countermeasures based on a browser extension and proposing a new concept to handle problems of fingerprint tracking.

In the phase of studying of the workflow of fingerprinting, the research had used attributes from Table 2-2 to create the hybrid fingerprinting technique, namely, object fingerprinting, canvas fingerprinting, plugin fingerprinting and font fingerprinting. The selected attributes for creating the fingerprinting ID mainly derived from studying code from fingerprintjs[35] and fybridjs[36]. The outcome assists in understanding how the fingerprinting technique can access the user data through attributes, how the fingerprinting technique can create the tracker (the fingerprint ID) and how the fingerprinting technique can track the user.

After understanding the basic knowledge of fingerprinting technique, the available fingerprint countermeasures were validated. According to Nikiforakis et al. (2014b) , a good fingerprint countermeasure should maintain balance between effectiveness of fingerprinting prevention and impact on the web browser while using it. Similarly, Torres et al. (2015) stated that the good fingerprint countermeasure should maintain the same relationship of attributes of a user computer after changing attributes in order to prevent the web browser standing out. Therefore, the preliminary experiment in this phase will be divided into four sections, namely, studying the effectiveness of fingerprinting prevention, studying the user experience, studying how to handle the attributes in order to inhibit the fingerprinting tracking and studying the information paradox.

The preliminary experiment would assist the research to understand the strengths and weaknesses of each countermeasure. These results of preliminary experiment assisted this research in finding limitations of existing countermeasures and extending knowledge

[35] https://github.com/Valve/fingerprintjs
[36] https://github.com/ammosh/fybridjs

about operation of existing countermeasures. These assisted the research in seeking a new concept to inhibit the fingerprint tracking.

4.4 Evaluation

According to Nikiforakis et al. (2014b), the evaluation of fingerprint countermeasure should consist of at least four aspects, namely, the effectiveness of fingerprinting prevention, the user's browsing experience, information paradox and performance overhead. Therefore, this phase would be split into three sections.

In order to evaluate the effectiveness of fingerprinting prevention, many fingerprint countermeasures are validated, comparing the effectiveness of fingerprinting prevention with the new fingerprint countermeasure of this thesis. The research simulates experiment similar to the tracking of fingerprinting companies as much as possible. The expected results should be that the new fingerprint countermeasure would show a high effectiveness of fingerprinting prevention. All details of this section are described in Chapter 7.

The ideal fingerprint countermeasure should have a negligible performance overhead. This stage will validate the performance overhead by observing the average execution time. The new fingerprint countermeasure will be validated by the average execution time in comparison with the unmodified browser. The result can reflect that the new fingerprint countermeasure has negligible side effects to running of the web browser.

In addition, the research had also added experiment on the information paradox. The fingerprint countermeasures will be validated in order to measure the level of information paradox through established metrics. The result will show which fingerprint countermeasures yields the highest level of information paradox.

In order to evaluate the user's browsing experience, the research began with defining the group of participants. All participants should have experience of using websites. Each group of participants would be arranged to use only one countermeasure. All participants will be asked to visit a chosen website and then the participant will answer the provided questionnaire after visiting the website. This research has applied pleasure-arousal-dominance (PAD) model from Huang et al. (2017), consisting of six metrics:

functional experience, hedonic experience, social experience, pleasure, arousal, dominance and word of mouth to design questionnaires. After collecting the data survey, the research will obtain the descriptive statistic which can use these data to assess the user satisfaction. The details of the survey are described in Table 4-2 below.

Table 4-2 details of survey

	Student satisfaction survey
Purpose	Evaluation
Group of participants	Undergraduate students
Survey Media	Written Survey
Survey question type	Closed-ended questions
Contact method	Personal contact
Administration method	Delivery and collection

4.5 Summary of this chapter

The research process has been described in this chapter. This process will be used for completing this thesis. A sequence of the research process will be designed according to the design science research process which consists of three main phases, problem identification, solution design and evaluation.

In consideration of the problem identification phase, the problem of user privacy had been raised, that is, how to protect the user from the fingerprint tracking. The research established that there were many possibilities that can inhibit the fingerprint tracking. Using browser extension appeared to be a good countermeasure against the fingerprinting technique. However, choosing a browser extension to prevent the fingerprint tracking still has limitations that needed to be addressed and developed further.

In the solution design phase, the available fingerprint countermeasures which claimed credit for preventing a fingerprinting technique will be validated for four aspects,

effectiveness of fingerprinting prevention, the user browsing experience, the performance overhead and the information paradox. The results of validation can reveal the strengths and weaknesses of each countermeasure which assisted the research to create a new fingerprint countermeasure.

In the evaluation phase, there are four main experiments to be conducted, evaluating the effectiveness of fingerprinting prevention, the user satisfaction, level of information paradox and the performance overhead. In order to evaluate effectiveness of fingerprinting prevention, the experiment would setup an environment similar to the fingerprint tracking work in the real world. Then, selected countermeasures would be validated by comparison with the proposed countermeasure. The results of comparison would disclose the effectiveness of fingerprinting prevention. As for the user satisfaction, the research would invite participants with skill in the use of computers to conduct the survey. The results of the survey would show which countermeasures have high user satisfaction. For evaluating the performance overhead of countermeasure, the proposed fingerprint countermeasure will be measured by the execution time in comparison with the unmodified browser. Finally, fingerprinting countermeasures will be measured the level of information paradox through metrics. The results will reveal the level of information paradox of tested countermeasures.

Chapter 5 The empirical study

In order to understand how the browser fingerprinting works, what is the main limitation of existing countermeasures and how a fingerprint countermeasure prevents the fingerprint tracking, all experiments have been classified as follows. This chapter starts with the study of how the browser fingerprinting technique works in practice in Section 5.1. The next section describes results of testing the developed fingerprinting code. In terms of testing fingerprint countermeasures, four aspects of existing fingerprint countermeasures had been tested, namely effectiveness of fingerprinting prevention, side-effects of the user browsing experience, how the fingerprint countermeasures handle the fingerprinting attributes, and a combination of fingerprinting attributes after changing attributes (the information paradox). The aim of all sub-sections in Section 5.3 is to find the strengths and weaknesses of each fingerprint countermeasure. The final phase would bring results to in-depth analysis.

5.1 Creating the hybrid fingerprinting website

According to Table 2-2, many fingerprinters have used a list of fonts, a list of plugins, JavaScript objects and canvas in HTML5 for fingerprinting the user computer. For this reason, the fingerprinting website should create four methods because the research is uncertain how many attributes can be prevented by the existing fingerprint countermeasures.

As shown in Table 2-1, Eckersley (2010) had reported the entropy values of a list of fonts and a list of plugins at 13.9 and 15.4 bits of entropy respectively. The experiments of Boda et al. (2012a) obtain entropy values of a list of fonts and a list of plugins as close to the experiments of Eckersley (2010). High entropy attributes can assist the diversity of fingerprinting ID. According to attributes used for fingerprinting in Table 2-2, both open source software for fingerprinting technique and fingerprinting companies used list of fonts and list of plugins to fingerprint the user computer. Thus, the research should include these attributes in the hybrid fingerprinting website.

In consideration of entropy value of fingerprinting based on JavaScript objects, many fingerprinters still use these attributes as a fundamental component to fingerprint the user computer. Even though many attributes in JavaScript objects have low entropy value, combining these attributes with high entropy attributes would contribute to creating a fingerprint shared with a very small subset of web browsers. Similarly, Mayer (2009) had used JavaScript screen and navigator object that can identify the web browsers uniquely, above 96%.

In recent years, there has been an increasing amount of literature on canvas fingerprinting. For example, Acar et al. (2014) had conducted a survey of canvas fingerprint technique by visiting the Top Alexa 100,000 websites. They found that 5.5% of websites use canvas fingerprinting to fingerprint a user computer. The result of the survey showed that many websites still use the canvas fingerprinting for tracking users. According to Laperdrix et al. (2017), their study demonstrated that the one the strength of canvas fingerprinting is stability and sort out canvas attribute as the most discriminating attribute. It is remarkably the second highest source of entropy when using it a mobile device.

Methods thought to be influencing the creation of fingerprint would be studied in this section. All previously described methods are a primary reason why this research should create the four methods because this research would like to study workflows of the fingerprinting technique used in practice.

In order to fully comprehend workflows of the fingerprinting technique, it cannot be ignored how the fingerprinting algorithm can perform and can track the Internet user. Typically, fingerprinting based on a list of fonts, fingerprinting based on a list of plugins, fingerprinting based on JavaScript objects and fingerprinting based on canvas in HTML5 have been deployed in the real world. Consequently, the Internet users have a high possibility of being tracked by means of four methods while navigating. Therefore, four methods of fingerprinting technique would be studied and developed into the hybrid fingerprinting website. The main purpose of development is to study the workflow of the basic fingerprinting technique, to observe side-effects of browsing the web, to study how the fingerprint countermeasures handle data and to notice the change of fingerprinting attributes after changing attribute values. The created hybrid fingerprinting website, ***http://www.pleasefingerprintme.org***, is revised from several open sources such as

FingerprintJS[37], Fybridjs[38], AM I unique[39], Darkwave Technologies[40] and FPdetective[41]. This site can produce a fingerprinting ID (value of fingerprinting) as a unique tracker. The produced fingerprinting ID remains unchanged even when the user computer is rebooted, or the browser is restarted or website URL is changed. Furthermore, the web cookies had been designed to check the change of fingerprinting ID after a user is fingerprinted by prompting a user at the web page.

5.1.1 **JavaScript Objects Fingerprinting**

The data of Internet users can be easily accessed through JavaScript objects while navigating the websites (Mayer, 2009). Until now, nearly all websites still use JavaScript for various reasons such as making visual effects on screen, computing data or running processes on web pages. On the client side, many web browsers are designed to support JavaScript. The web browsers had developed their own JavaScript engine based on the ECMAScript specification, a standard defined by ECMA[42], which affects each browser having different performance because of difference of JavaScript engine.

In the literature review, using JavaScript has been associated with the possibility to be fingerprinted. Normally, JavaScript is designed to provide various objects for a variety of reasons (Bielova, 2013). Each object consists of methods and variables. Both methods and variables of the JavaScript object can be accessed with a simple dot-notation. When the web browser runs a JavaScript object, methods and variables can be accessed and information sent back to a server. For this reason, using a JavaScript object is extremely helpful in getting access to the basic information of the user (e.g., screen resolution, plugins, browser, etc.) which lead to the use of obtained basic information to fingerprint. Since almost all websites currently use JavaScript on their web pages, this makes it extremely hard for the Internet users to recognize who is trying to track them. In this study,

[37]https://github.com/Valve/fingerprintjs2
[38] https://github.com/ammosh/fybridjs
[39]https://amiunique.org/
[40]http://www.darkwavetech.com/device_fingerprint.html
[41]https://github.com/fpdetective/fpdetective
[42] http://www.ecma-international.org/

many attributes of JavaScript objects were selected to create the identity (Fingerprinting ID) as listed below, more details a reason to choose each attribute provided in Appendix A.

- **Navigator (15 attributes):**
  UserAgent, appCodeName, appVersion, doNotTrack, product, productSub, cookieEnabled, vendor, vendorSub, online, platform, language, languages, mimeTypes, JavaEnabled;
- **Navigator.plugins (4 attributes):**
  Name, filename, description, length;
- ***Navigator.mimeTypes (4* attributes*)***
  enablePlugin, description, suffix, type;
- **Screen (11 attributes):**
  horizontalDPI, verticalDPI, availLeft, availTop, availHeight, availWidth, colorDepth, pixelDepth, width, height, bufferDepth;

There is a large volume of published studies describing attributes used for fingerprinting by using JavaScript. The research effort was to select suspicious attributes that could be selected by the fingerprinter companies in consideration of Table 2-2. The operating procedure of the object fingerprinting would directly access user data through selected JavaScript objects. The information accessed by this method will be computed into the user identification (Fingerprinting ID) after gaining the user data. Normally, almost all attributes were relatively straightforward to access, except plugin and mimeTypes. On the part of mimeTypes and plugins attributes, they were arrays of objects for which the method of connection is slightly difficult, compared with other attributes. The plugin object will be contained in the first array, and mimeType object will be contained in the second array. The plugin object consists of filename, description, length and name as mimeType contains attributes (e.g., description, enablePlugin, suffixes and type). The kind of content in mimeType is inevitably associated with the plugin. For example, the Chrome PDF Viewer was a plugin using application/pdf. Thus, some attributes related to PDF file would be contained in the mimeType object, shown in Figure 5-1.

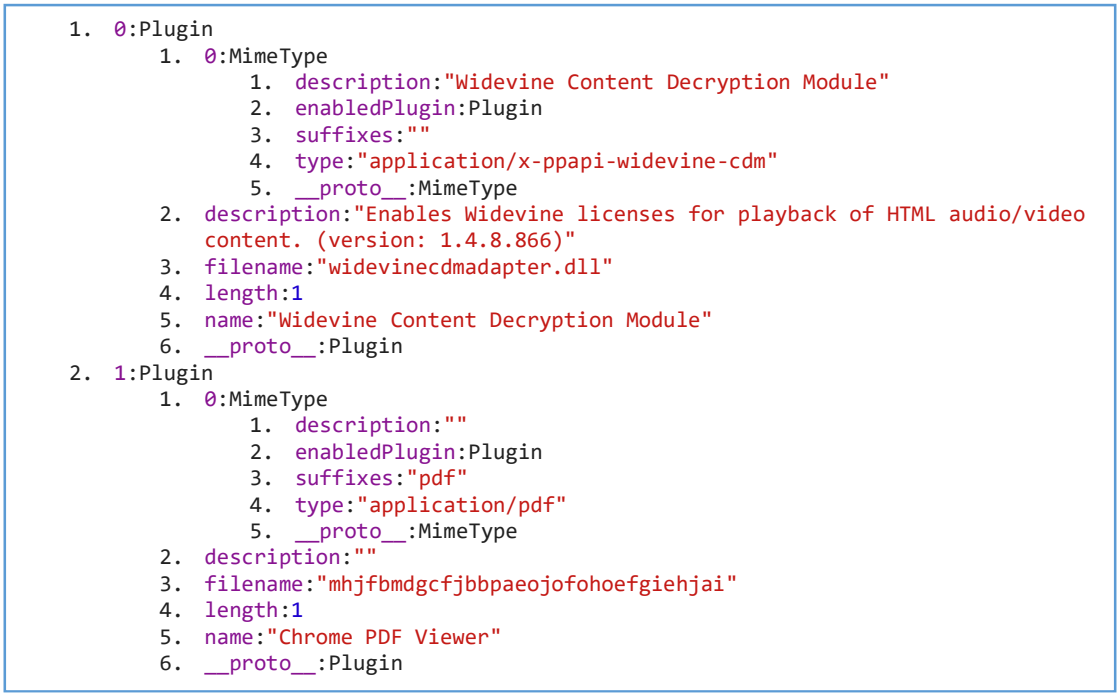


Figure 5-1 the relationship between plugin and mimeType

The JavaScript object fingerprinting was designed to access attributes, navigator and screen object. The accessed attributes will be used to compute fingerprinting ID. According to Table 2-2, several fingerprinters use various attributes to fingerprint the user. This research attempted to select the most feasible attributes being used by fingerprinters. The selected attributes would assist the research to understand how the fingerprint countermeasure handles attributes. In the case of detecting the object fingerprinting technique, some fingerprint countermeasures may regard connection with attributes given in Table 2-2 as an indicator of fingerprinting with the object fingerprinting technique.

5.1.2 **JavaScript Fonts Fingerprinting**

The font fingerprinting is a modern technique depending on the list of fonts stored on the user computer. The list of fonts is one attribute which has a high entropy, calculated by Eckersley (2010) at 13.9 bits of entropy. Unsurprisingly, almost all fingerprinting companies did not hesitate to utilize this attribute in their fingerprint algorithms with the aim of improving uniqueness of their fingerprinting ID (Nikiforakis et al., 2013). According to Englehardt and Narayanan (2016a), they found that the number of websites using fonts to fingerprint increased over seven-fold since 2013 after studying by Acar et al. (2013). In

addition, it is different from JavaScript object fingerprinting in terms of the method of accessing data, and details of the workflow of loading fonts through JavaScript described in Figure 5-3.

List of fonts on the user computer can usually be accessed through two methods, using plugin or JavaScript. Flash provided APIs for retrieving fonts of the user computer. Many fingerprinting companies (e.g., Bluecava, Iovation and ThreatMatrix) normally use the plugin APIs (e.g., Flash or Java, etc.) of the web browser to access the list of fonts as the first method (Nikiforakis et al., 2013). However, if any web browsers have not installed any plugins, they will use JavaScript to obtain the list of fonts as the contingency plan. The technique of using JavaScript is established by Nikiforakis et al. (2013), similar to stealing the CSS history (Jang et al., 2010), which it is capable of using to obtain a list of fonts even if a web browser disables its plugins.

To illustrate the workflow process of how to obtain the list of fonts with the assistance of JavaScript, an HTML document is usually comprised of a vast number of HTML elements of which each HTML element is designed with different purposes and uses. Font-family is one of the instructions used to specify the element's font. This instruction has a backup system in connection with many font names. If the font finding is not available, the font fingerprinting algorithm will find the next font in a fallback list as shown in Figure 5-2 below.

From Figure 5-2 below, the algorithm can implement this instruction to read the list of fonts from the user computer.

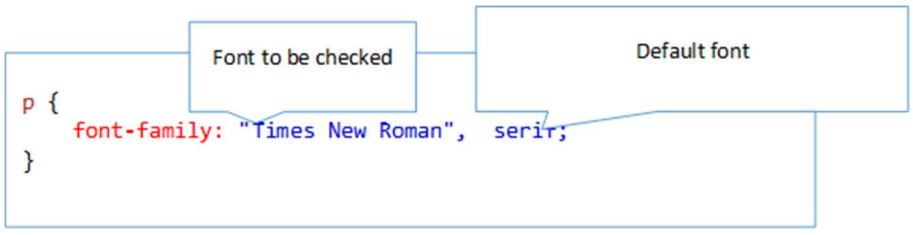


Figure 5-2 the checking of the current font and default font

From Table 5-1, supposing the algorithm is required to check whether Times New Roman is stored on the user computer or not, Font-family will be used to validate between Times New Roman and default font. If the fingerprinting algorithm could obtain values of offsetWidth and offsetHeight as equal to the value of offsetWidth and offsetHeight of the default font, the algorithm will determine that Times New Roman is unavailable on the user computer. In contrast, if Times New Roman font is stored in the user computer, the obtained value of offsetWidth and offsetHeight will be dissimilar from offsetWidth and offsetHeight of the default font. By doing this repeatedly, this algorithm will obtain a list of almost all fonts in the user computer through the loop instruction.

Table 5-1 the difference of offsetWidth and OffsetHeight of font

Fonts	**Offset Width(pixel)**	**Offset Height(pixel)**	**String**
Default	1340	80	fingerprinting
Sans-serif	1445	82	fingerprinting
Times New Roman	1544	81	fingerprinting

This technique will systematically find fonts. The number of fonts checked on the user computer is derived from original fingerprintjs library[43], around 460 fonts - Windows: 231 fonts, Mac OS: 167 fonts, and other operating systems: 62 fonts. However, this method can detect only popular fonts. The unpopular fonts are undetectable because they are not stored in the list (Nikiforakis et al., 2013). The basic workflow of getting a list of fonts is described in Figure 5-3.

The mechanism behind the font fingerprinting consisted of loading fonts from HTML element and comparing the offsetWidth and offsetHight of a font. For this reason, some fingerprint countermeasures acknowledge this effort as an index of fingerprinting user by using the font fingerprinting technique.

[43] https://github.com/Valve/fingerprintjs2

Algorithm: access the list of fonts

```
Output: list of fonts
Declare Boolean available
Declare Array fonts
Declare Array available_fonts
Declare Array default_font
Set fonts["Arial","cursive","monospace", … ,  460 fonts]
Set default_font["sans-serif"]
for i=0 to default_font.length step 1 do
    reading offset of width and height of default font
end for
for i=0 to fonts.length step 1 do
Set available :=  false
  for j=0 to default_font.length step 1 do
      reading font in a list
      reading offesetWidth and offesetHeight of font from a list
        if (offsetWidth and offesetHeight) ≠ (offsetWidth and offesetHeight of default fonts)
            push font into available_fonts
         end if
   end for
end for
```

Figure 5-3 pseudocode of loading fonts through JavaScript

5.1.3 **Plugin fingerprinting**

A plugin is a crucial factor that is found to be influencing the uniqueness of fingerprinting ID (Boda et al., 2012a, Eckersley, 2010). Despite the many benefits of using a

plugin, questions have been raised about the safety of using a plugin. The Internet users are unaware that installed plugins on their web browsers do not only assist them to surf websites conveniently but make it easier to distinguish their identity. Typically, plugins will be customised in accordance with the requirement of an individual user. Access to this attribute is extremely helpful for fingerprinter companies or someone who desires to use this attribute for fingerprinting. According to experimental results of Eckersley (2010), the list of plugins is a high-entropy attribute. Using only a list of plugins has sufficiently high entropy (15.4 bits) to generate a 1-in-43,237 fingerprint.

Algorithm: access the list of plugins

```
Output: list of user's plugins
Declare String current_plg
Declare String plgs
Declare Array props[]
Set props['name','description','filename']
Set plgs : = " "
Total_plugins := count the number of plugins
for i= 0 to Total_plugins step 1 do
        for j = 0 to number of props step 1 do
                current_plg += (props[j] +navigator.plugins[i][props[j]])
        end for
   plgs += current_plg
end for
```

Figure 5-4 pseudocode of loading list of plugins

A fingerprinting company just selects the rest of the attributes which may be low-entropy attributes (e.g. operating system, screen resolution, etc.) for creating a fingerprinting ID with sufficient stability. The instruction to exhibit the list of plugins is

navigator.plugins.length which can reveal name, description and filename as shown in Figure 5-4.

The procedure of plugin fingerprinting is the call of *navigator.plugins.length* and repeated loading of plugins before displaying a web page. In view of all that has been mentioned so far, one may conclude that a fingerprint countermeasure will determine the call of *navigator.plugins.length* and repeated loading of plugins as the indicator of being fingerprinted by the plugin fingerprinting technique.

5.1.4 **Canvas fingerprinting**

Canvas fingerprinting is a novel technique which records a fingerprinted computer by using canvas element in HTML5. Identifying with HTML canvas element is accurate enough to distinguish the identity of a user (Mowery and Shacham, 2012). Their experiments can observe 116 unique fingerprinting values, with a sample entropy of 5.73 bits. Typically, HTML5 element possesses an attribute to draw the graphic contents. This attribute is called the canvas element. The canvas possesses writing and reading methods which can be exploited by fingerprinters. Fingerprinting by canvas element is straightforward. It firstly draws an arbitrary context on the web page through the canvas element by using a writing method of the canvas.

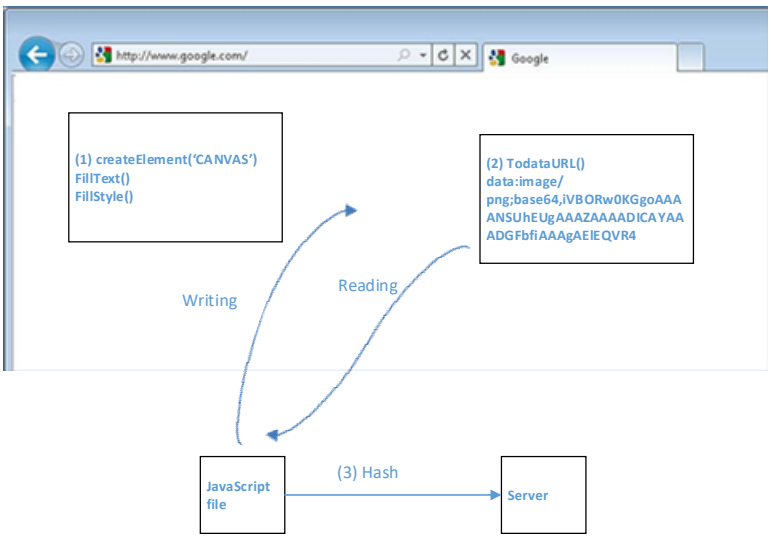


Figure 5-5 the operation of canvas fingerprinting

The image data (pixel) on the canvas element then will be read by using a reading method, toDataURL(), of the canvas. The collected image data will depend on the type of

operating system and web browser used by the user. Thus, the difference of collected image data can be implemented as a tracker for tracking the user's browsing habits (Mowery and Shacham, 2012). The workflow is shown in Figure 5-5.

From the point of view of fingerprint countermeasures, they establish that the call of reading and writing through the instruction of canvas in HTML5 is fingerprinting user by means of the canvas fingerprinting technique.

5.2 Implementation to website

To date, various methods have been developed and introduced to track a user. For this research, most fingerprinting source code that tends to be used most recently would be applied in this research. The research used various methods of fingerprinting, such as AmIunique[44] or fingerprintingjs2[45], for fingerprinting a user computer with multiple methods. This research attempts to imitate the operation of fingerprinting technique similar to an open source software implementation of fingerprinting technique. The developed hybrid fingerprinting code will be uploaded on the server, http://www.pleasefingerprintme.org.

In the beginning, the hybrid fingerprinting website[46] will extract user data via attributes that can be obtained from a visit of the web browser by a user. Each method of fingerprinting technique will fingerprint the user computer in multiple ways and then show its fingerprinting ID to the screen with the hybrid fingerprint ID derived from combining four methods of fingerprinting.

To consider Figure 5-6, the developed hybrid fingerprinting website can be separated into two parts, Gatherer and Creator. Gatherer collects the browser's attributes (or properties) by using JavaScript, as Creator computes the browser's gathered attributes into a fingerprinting ID.[47]

---

[44] https://amiunique.org/
[45] https://github.com/Valve/fingerprintjs2
[46] http://www.pleasefingerprintme.org/

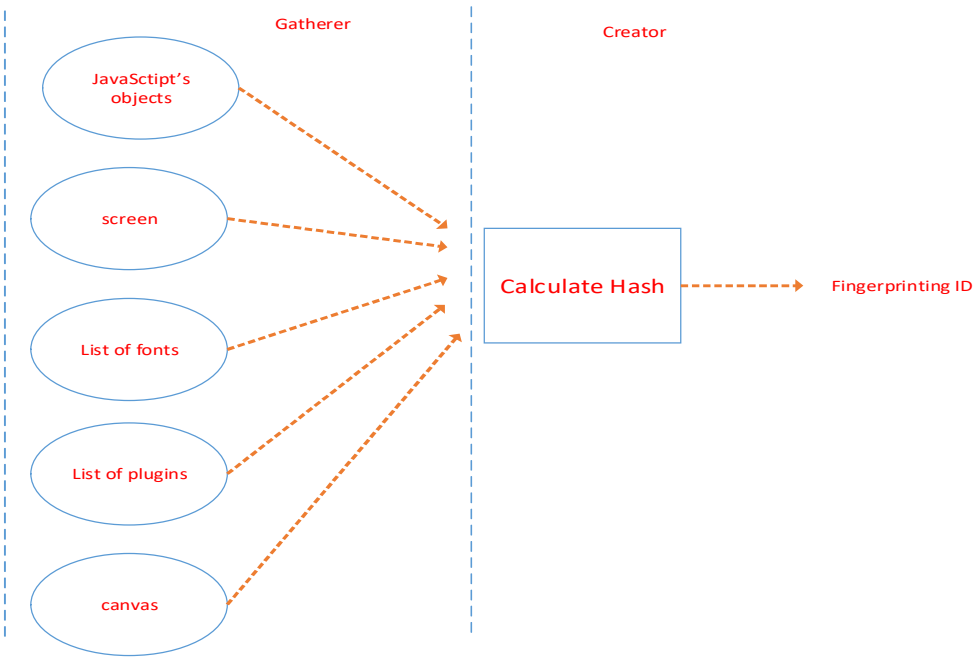


Figure 5-6 overview of hybrid browser fingerprinting

The hybrid fingerprinting website used various attributes on the web browser to create the fingerprinting ID which are clarified below.

JavaScript object: 34 attributes were accessed through navigator, screen, offset. The Gatherer queries 34 attributes and then concatenates 34 attributes in the array and regard them as one attribute.

JavaScript fonts: Gatherer will run a procedure as demonstrated in Figure 5-3 for detecting fonts on the user computer. Gatherer will use 460 fonts, consisting of Mac: 167 fonts, Windows: 231 and other: 62 fonts, from Bluecava[48] for checking the presence of fonts on the user computer. The detected fonts will be used for fingerprint.

Plugins: Gatherer will run the procedure as demonstrated in Figure 5-4. The obtained value will be used as one attribute in order to fingerprint a user by means of the plugin fingerprinting technique.

Canvas: This approach will apply this method by first drawing a pangram on the canvas element and then retrieve the drawn image data as a base64 encoded string.

[48] http://bluecava.com/

Site developing contained four different methods of fingerprinting. When a user visits the site and then clicks "*Testing*", the number of collected attributes will be extracted as shown in Table 5-2.

Table 5-2 the number of collected fingerprinting attributes

Method	**Number of attributes**
JavaScript Objects Fingerprinting	34 attributes
List of Fonts Fingerprinting	1 attribute
List of Plugins Fingerprinting	1 attribute
Canvas Fingerprinting	1 attribute
Total	37 attributes

5.2.1 **Result of testing hybrid fingerprinting website**

Traditionally, the browser fingerprinting technique consists of two characteristics, namely the stability and diversity. As mentioned in Chapter 2.4, the diversity is the ability of browser fingerprinting technique to be able to produce a unique fingerprinting ID, no other devices having the same fingerprinting ID. However, the research did not pay attention to the diversity of fingerprinting ID as it is not related to the purpose of this research which required to only study workflows of the browser fingerprinting technique. For this reason, the research focuses only on the process of creating the fingerprinting ID of each method. The site was designed to produce five fingerprinting IDs: hybrid fingerprinting ID, font fingerprinting ID, plugin fingerprinting ID, JavaScript Object fingerprinting ID and canvas fingerprinting ID. The reason was that the research required to know how many methods of fingerprinting can be prevented by a fingerprint countermeasure.

The aim of the research is to explore a concept based on the browser fingerprinting technique. Thus, the research will focus only on created fingerprints that can work normally when using a basic fingerprinting technique. It means that a created fingerprint

can create an identity and can track a user with obtained information through a web browser.

During tracking across websites, the research had embedded fingerprinting code in three websites, maneerotehotel.com, sodaresort.com and rmuti.ac.th, and then compelled the web browser to visit three websites as shown in Figure 5-7. The result of fingerprinting ID visiting three websites shows still identical fingerprinting ID which the fingerprinting server can distinguish when a web browser visits a fingerprinting site. It showed that hybrid fingerprinting website can track a user across websites by means of the fingerprinting technique.

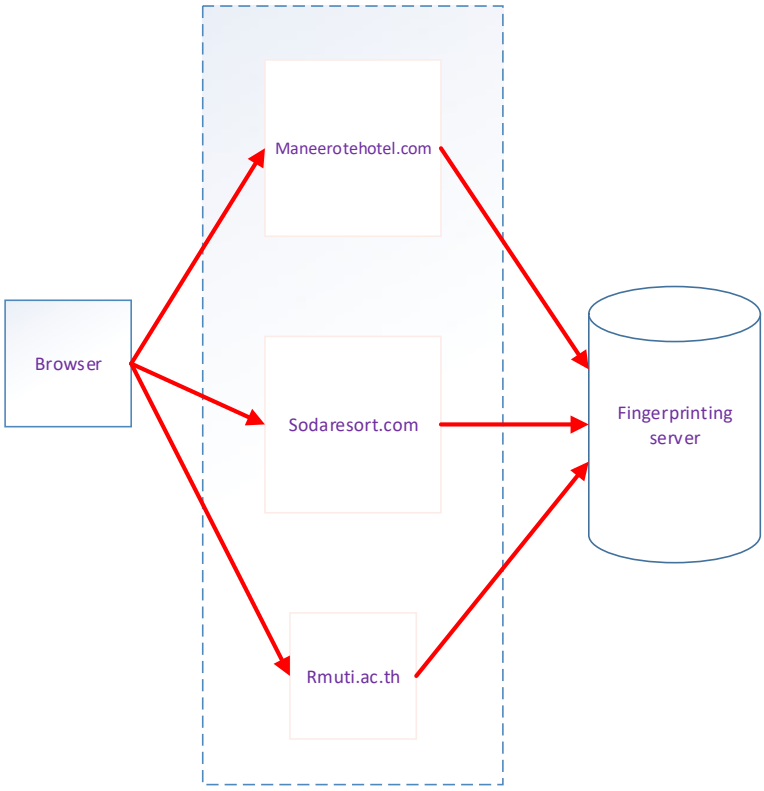


Figure 5-7 testing fingerprinting code

Even though this set of experiments is relatively small, it remains large enough to take some conclusions: that the created hybrid fingerprinting website can track the user computer. In addition, the research did not need to test entropy value of attributes because many published studies had already made this experiment and results of entropy showed similar results, as shown in Table 2-1.

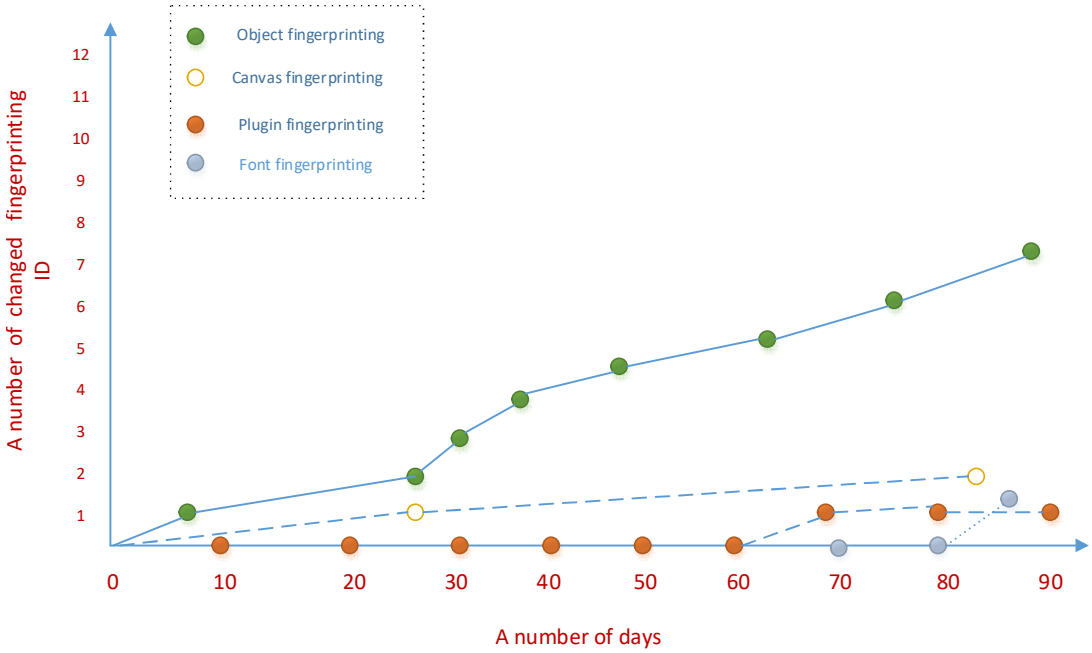


Figure 5-8 a number of changed fingerprinting ID

Given the Figure 5-8, the graph illustrates a change of four fingerprinting ID over 90 days. It can be clearly seen that fingerprinting based on object JavaScript shows the most frequent change of fingerprinting ID, about twice a month. The rest of fingerprinting technique, canvas fingerprinting, font fingerprinting and plugin fingerprinting, seems to change only once every three months.

This graph clearly illustrates that list of fonts, canvas, plugins are so attractive for fingerprinters as an often unchanging attribute. It implies that fingerprinters could develop an algorithm to anticipate the gradual change of these attributes with the purpose of maintaining the stability of their fingerprint. Despite the fact that the fingerprinting ID based on object JavaScript changes the most, the change of fingerprinting ID can be anticipated because most of changed attributes correspond to update on the version of the web browser.

5.3 Testing of existing countermeasures

The main purpose of this section is to develop an understanding of limitations of existing fingerprint countermeasures based on a browser extension. This section

conducted preliminary experiments on almost all aspects of fingerprint countermeasures. From the literature review in Chapter 3, the current fingerprint countermeasures have three main problems: inability to block fingerprinting (FaizKhademi et al., 2015), side-effects of the user browsing experience (Nikiforakis et al., 2014b), and the information paradox as a result of changing attributes (Eckersley, 2010). This study aimed to address the following research question: "*How can randomized attributes be introduced into a web browser in order to prevent browser fingerprinting with a minimal impact on the browsing experience?*". For this reason, the methodological approach taken in this study is developed based on problems encountered in a literature review. Thus, this section would be separated into three main parts, namely testing ability to prevent the fingerprint, testing the user experience resulting from use of a fingerprint countermeasure, testing the information paradox after changing attributes. All sections are designed to find the limitation of existing countermeasures in order to prevent the fingerprint tracking effectively in the future.

5.3.1 **Testing prevention**

The research design had simulated scenarios similar to the browser fingerprinting technique used in practice for testing inhibition of tracking this technique. Typically, when a user changes to another website, if a user is fingerprinted by the browser fingerprinting technique, the fingerprinting ID would show the same fingerprinting ID. On the other hand, if a user is not tracked by the fingerprinting technique, the fingerprinting ID would show different fingerprinting ID. For this reason, the research had created 30 virtual fingerprinting sites to test the change of fingerprinting ID. The created virtual fingerprinting site would be placed by the fingerprinting script. If a user visits any virtual fingerprinting site, a user is tracked immediately with fingerprinting script placed in a virtual site. Meanwhile, if any fingerprint countermeasures can change the fingerprinting ID while changing URL of sites, it signifies that the tested countermeasure can prevent the fingerprint tracking. Details of experimental setup are shown in Figure 5-9.

- ***Experimental setup***

In practice, the fingerprinting technique would contain fingerprinting script on first-party websites. Using this method, the fingerprinting script was delivered through advertising companies. When the user visits a first-party website using an advertising

network of a fingerprinting service provider, the fingerprinting ID would be sent back to the fingerprinting service provider. The first-party website may not even be aware that users are being fingerprinted by another person.

The research had designed a similar scenario as the fingerprinting technique working in practice, for testing inhibition of the tracking as mentioned above. Typically, if a user changed to another website, the fingerprinting ID should be the same fingerprinting ID in case that the user is tracked with the fingerprinting ID. For this reason, the research had created the virtual fingerprinting site to test the changing of fingerprinting ID as in Figure 5-9.

Figure 5-9 the virtual website

With reference to Figure 5-9, the research had installed Ubuntu-16.04.2-desktop-amd64 on a desktop machine, with 16 GByte of RAM, an Intel Core i7-4770 CPU and a clock speed of 3.40 GHz. Then it configured 30 virtual websites, each site embedded with the hybrid fingerprinting script. More details of the configuration of a virtual website are derived from this URL[49]. This preliminary experiment used Firefox version 53.0 and Chrome version 57.0.2987.133 for testing each countermeasure related to Table 5-3. A web browser will then be forced to visit each virtual website until 30 virtual web sites are visited.

[49] https://www.digitalocean.com/community/tutorials/how-to-set-up-apache-virtual-hosts-on-ubuntu-14-04-lts

Table 5-3 Lists of Existing countermeasures

Countermeasure	Software	Average satisfaction rating	Installations
Rubberglove[50]	Running on Chrome	4/5	546
Chameleon[51]	Running on Chrome	Open source software	
CanvasFingerprintBlock[52]	Running on Chrome	3/5	7,474
ChromeDust [53]	Running on Chrome	Open source software	
StopFingerprinting[54]	Running on Chrome	2.5/5	840
UserAgent Switcher for Chrome[55]	Running on Chrome	4/5	1,269,487
Canvas Fingerprinting Blocker[56]	Running on Firefox	2/5	2,480
FireGloves[57]	Running on FireFox	Open source software	
FP-Block [58]	Running on FireFox	0/5	70
Stop fingerprinting [59]	Running on FireFox	3.5/5	1759

(*) - *scores of average satisfaction rating and installation deriving from a web store of each web browser.*

[50] https://chrome.google.com/webstore/detail/rubberglove/koabfojebhfdjnligkcihoeekimoekpg?hl=en
[51] https://github.com/ghostwords/chameleon.
[52] https://chrome.google.com/webstore/detail/canvasfingerprintblock/ipmjngkmngdcdpmgmiebdmfbkcecdndc.
[53] https://github.com/tgalvin13/ChromeDust
[54] https://chrome.google.com/webstore/detail/stopfingerprinting/kfhlgmfkolojpnmhgggilmillpcokmnb?hl=en
[55] https://chrome.google.com/webstore/detail/user-agent-switcher-for-c/djflhoibgkdhkhhcedjiklpkjnoahfmg
[56] https://addons.mozilla.org/en-gb/firefox/addon/canvasblocker/
[57] https://fingerprint.pet-portal.eu/?menu=6
[58] https://addons.mozilla.org/en-GB/firefox/addon/fp-block/
[59] https://addons.mozilla.org/en-GB/firefox/addon/stop-fingerprinting/

As regards current fingerprint countermeasures, many are available on the Internet. They were gathered so that the research can bring these fingerprint countermeasures to test the fingerprinting prevention. The web browser will be installed with each countermeasure as Table 5-3, and then it will be compelled to visit all virtual websites containing the fingerprinting script. Furthermore, some open source software that claimed that they are able to prevent the browser fingerprinting technique would also be tested in this section.

While few papers , such as FPGuard (Weldemariam, 2015) and Privaricator (Nikiforakis et al., 2014b), claims that their countermeasures can inhibit the fingerprint tracking, these unproven claims are more difficult to evaluate whether their model can prevent better or not as the status of software has not been made available. Therefore, in the beginning, this research had mainly focused only on existing countermeasures which can now be downloaded through the Internet.

- **How to observe fingerprinting prevention**

In practice, there are a few methods to find the effectiveness of fingerprint countermeasures that can prevent the fingerprint tracking. However, this research had determined two metrics to measure the effectiveness of fingerprinting prevention as follows.

- **Unchanged fingerprinting ID**: The fingerprinting ID will not be shown on the screen the fingerprint countermeasure can block the fingerprint tracking.
- **Change of fingerprinting ID**: If the fingerprinting ID is altered on every visit to the fingerprinting website, this means that the countermeasure can prevent fingerprint tracking.

In order to collect the percentage effectiveness score of fingerprinting prevention, the web browser is forced to visit each virtual fingerprinting website until completion of visiting 30 virtual fingerprinting websites, observing the change of fingerprinting ID. Each virtual fingerprinting website was designed with different URLs in order to support metrics as mentioned above. The research would compare consecutive fingerprinting ID. For example, if the first fingerprinting ID was similar to the second fingerprint ID, the data would store ‘0’ in the database - meaning the countermeasure was unable to prevent

tracking. In contrast, if the first fingerprinting ID was dissimilar to the next fingerprint ID, the data would store '1' - meaning the countermeasure was able to prevent the fingerprinting tracking. Next, the third fingerprinting ID would be compared with the second fingerprinting ID and the same procedure would proceed until completion of the visit. In the case that the fingerprint is not showing up on the screen, the data would store '1' in the database.

For how to determine the effectiveness of fingerprinting prevention, if any countermeasures showed a small number of the same fingerprinting ID or every fingerprinting ID unique for every visit to the virtual website, it signified that the tested fingerprint countermeasure had a high effectiveness of fingerprinting prevention.

- **Result of measuring the effectiveness of fingerprinting prevention**

Table 5-4 shows the result of fingerprinting effectiveness of each countermeasure with percentage calculation. The number clearly showed that many countermeasures were unable to prevent four methods of fingerprinting technique except RubberGlove and Fireglove. Some fingerprint countermeasures, such as Chameleon, CanvasFingerprintBlock, ChromeDust and Stop fingerprinting, show 0% of inhibiting four methods of fingerprinting technique, unable to prevent the fingerprint tracking as claimed.

With regard to RubberGlove and Fireglove, they can prevent four methods of fingerprinting technique as the fingerprinting ID was not showing up on the screen – it meant that they could inhibit the fingerprint tracking as they claimed.

The remaining fingerprint countermeasures showed the inhibition of two methods of fingerprinting technique. Notably, the FP-Block showed 93.3 percent of inhibiting the JavaScript Object fingerprinting. Similarly, Stop fingerprinting showed 90 percent of inhibiting the font fingerprinting. UserAgent Switcher for Chrome showed 100 percent of inhibiting JavaScript object fingerprinting, as well as Canvas Fingerprinting Blocker showed 100 percent of inhibiting canvas fingerprinting. Unsurprisingly, the hybrid fingerprinting ID would be changed every time if any method of fingerprinting is changed because hybrid fingerprinting ID used four methods of fingerprinting technique to create the hybrid fingerprinting ID.

Table 5-4 result of producing the different fingerprinting ID

Countermeasure	Hybrid fingerprinting	Object JavaScript (navigator, screen)	List of fonts	List of plugins	Canvas
RubberGlove	*	*	*	*	*
Chameleon	0%	0%	0%	0%	0%
CanvasFingerprintBlock	0%	0%	0%	0%	0%
ChromeDust	0%	0%	0%	0%	0%
StopFingerprinting	0%	0%	0%	0%	0%
UserAgent Switcher for Chrome	100%	100%	0%	0%	0%
Canvas Fingerprinting Blocker	100%	0%	0%	0%	100%
Fireglove	*	*	*	*	*
FP-Block	100%	93.3%	0%	0%	0%
Stop fingerprinting	100%	0%	90%	0%	0%

(*) - Fingerprinting ID is not showing

Other countermeasures are incapable of inhibiting the fingerprinting tracking as they claimed. These countermeasures would be removed from the next experiment due to ineffectiveness to inhibit the fingerprint tracking.

To sum up, regarding only fingerprinting prevention, many countermeasures cannot prevent fingerprint tracking as they claimed. Possibly, some countermeasures might use the concept of Tor Browser that is difficult to measure with this experiment or some functions written on the browser extension may be deprecated. In addition, the research also established that some countermeasures cannot prevent any methods of fingerprint tracking.

5.3.2 Side effects of fingerprinting prevention

This experiment has been divided into three sub-experiments. In the beginning, it will find which countermeasures show side effects on the web browser, followed by finding which methods are used to change attributes, and then finding which attributes have a strong impact on a web browser.

5.3.2.1 Studying side effect of countermeasures

A user's browsing experience is an important factor that affects users. Even though some countermeasures can prevent the fingerprinting tracking, it did not mean that a web browser will not be affected. Only preventing the fingerprinting tracking is insufficient for the good fingerprint countermeasure. It should be effective in fingerprinting prevention in conjunction with insignificant side-effects to the web browser (Nikiforakis et al., 2014b).

Experimental setup

As discussed above, this section studied adverse impacts of existing countermeasures to the web browser by selecting only countermeasures that can prevent the fingerprint tracking from the result of Table 5-4, RubberGlove, UserAgent Switcher for Chrome, Canvas Fingerprinting Blocker, Fireglove, FP-Block and Stop fingerprinting, to study further. Each countermeasure will be installed on Firefox version 53.0 and Chrome version 57.0.2987.133 for testing the user browsing experience. A web browser will then be forced to visit a website for the purpose of monitoring the side effects of use.

User experience is the level of emotion, positive and negative, that a user experiences during and after product use (Schulze and Krömker, 2010). They define user experience as consisting of relatedness (being fast and easy), stimulation (balancing feeling of success), competency (avoiding stress and reducing complexity), popularity (adopting different roles), autonomy (having data control), and security (feeling secure). This research will omit popularity and autonomy as they are not consistent with this work. The remaining attributes will be distributed into four metrics in order to assess six fingerprint countermeasures without any bias, according to Schulze and Krömker (2010). These metrics are: The Display Problem (stimulation), The Functionality Problem (competence), The Difficulty of Use (relatedness), and The Login Problem (security), details shown in Table 5-5.

Table 5-5 Metrics for assessing the user browsing experience

Metric	Title	Description
Metric 1	The Display Problem	Content, fonts, and screen sizes are modified
Metric 2	The Functionality Problem	Video and sound are unavailable, or some functionality does not work properly
Metric 3	The Difficulty of Use	Browser runs slowly, and it is tough to navigate sites
Metric 4	The Login Problem	The user faces difficulties with website login

Regarding the existing published research on the fingerprinting prevention, there is no standard that mentions how to measure the user browsing experience. In the beginning of the preliminary study, the research had determined four metrics as mentioned above, with the aim to measure the user browsing experience.

How to assess the user browsing experience

Regarding the method to collect the result of user browsing experience, the web browser with the installed fingerprint countermeasure would be forced to visit the selected websites. The user experience of using each fingerprinting countermeasure would be observed by three users. If any tested countermeasure showed the same problem more than two times, the research would determine that this tested countermeasure has a problem of the user experience. For example, if the tested countermeasure showed the problem of display for more than two times, the research would assume that the tested countermeasure had a problem with metrics of problem of the display.

Result of measuring the user browsing experience

Comparing the two results between Table 5-4 and Table 5-6, it can be seen that a countermeasure that could prevent fingerprint tracking effectively in Table 5-4 is not necessarily always a good countermeasure. From the preliminary experiment, many countermeasures have troubles with the user browsing experience. Table 5-6 showed

results of user experiences of each metric visiting websites three times with percentage. It can be clearly seen that each countermeasure had shown different side effects to the web browser.

RubberGlove and Fireglove showed the most substantial impact on the web browser. The second most substantial impact is from FP-Block, followed by UserAgent Switcher for Chrome. On the other hand, Stop Fingerprinting and Canvas Fingerprinting showed the least significant impact on the web browser.

With respect to Table 5-6, FP-Block is the only countermeasure that considers the user browsing experience. In the concept of FP-Block (using web identities), it used only prevention part. In terms of implementation, it is contradictory, using both detection part and prevention part. Usually, the concept of using detection part was to find a suspicious code loading suspicious attributes and then find a proper way to prevent the fingerprint tracking consistent with detection of suspicious code. FP-Block would find only a loading of suspicious attributes and then inform users through a web page that they may be tracked by the fingerprinting technique. Users needed to choose options, blocking or allowing, every time to visit a new website, new page within the same URL or refreshing a web page. This was annoying to three users with 66.6 percent of problems of display. Particularly, as some websites had a sub domain that it appeared FP-Block assumed to be a different website, it continued to warn the user. Frequent notifications about tracking appeared to be a substantial problem for FP-Block because it showed a warning for almost all websites that attempted to track a user with the fingerprinting technique.

Overall, the obtained results are not sufficient, just showing side effects, to determine why each countermeasure shows different side effects to the web browser. Therefore, it is necessary to investigate further a way to change attributes of each countermeasure.

Table 5-6 the impact of browser to the user experience

Countermeasure	Problems of display	Problems of functionality	Difficulty to use	Login problem
RubberGlove	100%	100%	100%	100%
UserAgent Switcher for Chrome	100%	0%	100%	0%
Canvas Fingerprinting Blocker	66.6%	0%	0%	0%
Fireglove	100%	100%	100%	100%
FP-Block	66.6%	66.6%	66.6%	0%
Stop fingerprinting	0%	0%	0%	0%

5.3.2.2 Studying a method of changing attributes

Previous testing, Table 5-6, have reported the side effects on the web browser while visiting a site, but it is still insufficient to understand why each countermeasure shows different side effects and what is the mechanism behind alteration of attributes.

Experimental setup

In order to determine the cause of side effects, the research had established two scenarios as follows. They would disclose a method and a number of changed attributes.

1. **First scenario**

   Each countermeasure will be installed on a web browser and then it will be forced to visit the hybrid fingerprinting websites[60] 30 times in order to observe change of attributes. If an attribute is blocked, it will not be displayed on the screen. Likewise, if an attribute is changed by the randomization technique, the attribute value would be

[60] http://www.pleasefingerprintme.org

changed every time a web browser visited. However, this scenario cannot detect when any countermeasure uses a spoofing technique.

2. Second scenario

Each countermeasure will be installed on a web browser and then it will be forced to visit the 30 virtual fingerprinting website[61] as shown in Figure 5-9. If any attributes change when changing a URL of a site, it means that the tested countermeasure uses a spoofing technique.

How to assess the user browsing experience

If attribute shows undefined value more than 15 times every time while visiting a hybrid fingerprinting website, it means that the tested countermeasure uses a blocking technique.

If attribute shows different values more than 15 times every time while visiting a hybrid fingerprinting website, it means that the tested countermeasure uses a randomization technique.

If attribute shows different values more than 15 times every time while visiting virtual hybrid fingerprinting websites, it means that the tested countermeasure uses a spoofing technique.

Result of observing the change of attributes

Considering both Table 5-6 and Table 5-7, it can be clearly seen that a number of handled attributes are related to the side-effects on a web browser. If any countermeasures increased the number of attributes to be handled, the web browser had a high probability to have increasing side-effects, comparable to a countermeasure choosing a small number of attributes. For example, RubberGlove and Fireglove change many attributes (16 attributes) more than Stop fingerprinting (2 attributes). Thus, they have a high probability to produce more side effects on a web browser, as opposed to Stop fingerprinting changing only two attributes.

Regarding a method to handle attributes, the countermeasures using blocking technique seem to show a substantial impact to the web browser, as opposed to the countermeasures

[61] http://www.pleasefingerprintme.org

using spoofing technique or randomization technique. For example, RubberGlove uses the blocking technique to inhibit the fingerprinting tracking. The result from Table 5-6 shows that RubberGlove shows substantial side effects on the web browser. Therefore, undoubtedly the blocking technique causes a significant impact on the browsing experience.

Table 5-7 method to inhibit the fingerprinting technique

Attributes	**RubberGlove**	**UserAgent Switcher for Chrome**	**Canvas Fingerprinting Blocker**	**Fireglove**	**FP-Block**	**Stop fingerprinting**
List of plugins	Blocking	-	-	Blocking	Blocking	Blocking
List of fonts	Blocking	-	-	Blocking	-	Randomization
User-agent	Blocking	Spoofing	-	Spoofing	Spoofing	-
HTTP header Accept-Language	-	-	-	-	Spoofing	-
appCodeName	Blocking	-	-	-	Spoofing	-
Product	Blocking	-	-	-	Spoofing	-
Product-Sub	Blocking	-	-	-	Spoofing	-
platform	Blocking	-	-	randomization	Spoofing	-
Online	Blocking	-	-	-	Spoofing	-
appVersion	Blocking	-	-	spoofing	Spoofing	-
cookiesEnabled()	Blocking	-	-	-	Spoofing	-
javaEnable()	Blocking	-	-	-	Spoofing	-
Navigator.mimeType ()	Blocking	-	-	Blocking	Spoofing	-
Screen color and pixel depth	-	-	-	-	-	-
Screen width and height	-	-	-	spoofing	-	-
Screen availLeft, availTop, availHeight and availWidth	-	-	-	-	-	-
Screen hosizontalDPI, verticalDPI	-	-	-	-	-	-
Do not track	Blocking	-	-	-	Spoofing	-
CPU	Blocking	-	-	-	Spoofing	-
Language	Blocking	-	-	randomization	Spoofing	-
Languages	Blocking	-	-	Spoofing	Spoofing	-
Canvas element	-	-	Randomizing	•	Blocking	-
Timezone	-	-	-	Spoofing	-	-

In term of selecting entropy value, most countermeasures needed to choose high entropy value for changing the user identification. Notably, each countermeasure selected different high-entropy attribute to prevent the browser fingerprinting technique. The difference of choosing attributes had raised doubts about which attributes have a considerable influence on side effects on the web browser.

To sum up, an increasing number of attributes is associated with escalating side-effects on the web browser. While using blocking technique seemed to create a substantial impact on the user browsing experience, spoofing and randomization showed insignificant impact to the user browsing experience. Finally, most countermeasures choose high entropy attributes to change the user identification (Fingerprinting ID).

3. Testing side-effects of changing an attribute

The previous preliminary experimentation has shown the relationship between changing attributes and side effects on the web browser. However, the evidence for this relationship is inconclusive because the result showed only the side effects of changing an attribute. It is not showing which attributes are a substantial influence on the display of a web browser.

This experiment endeavours to change attributes on the user computer by using getters[62], as established in ECMAScript5. The JavaScript object defined by using getters was injected in a web page by using a browser extension. It means that the web browser will change the original values before running a web page. The main purpose is to find which attributes affect the display of web page.

Because there are many attributes on the user computers, this experiment divides attributes into two groups, low entropy and high entropy. At first the experiment selected high entropy attributes and then changed each attribute value by visiting a website until it can see the side effects of changing attributes. Similarly, the low entropy attributes will be changed from their original values and the consequence of changing attributes will be observed.

[62] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/get

Table 5-8 side effects of changing attributes

Attributes	Problems of display	Problems of functionality	Difficulty to use	Login problem
List of plugins	Low	Low	-	-
List of fonts	-	-	-	-
User-agent	High	High	High	-
HTTP header Accept-Language	High	Low	-	-
appCodeName	-	-	-	-
Product	-	-	-	-
Product-Sub	-	-	-	-
platform	Low	Low	Low	-
Online				
appVersion	-	-	-	-
cookiesEnabled()	-	-	-	-
javaEnable()	-	-	-	-
Navigator.mimeType ()	Low	Low	-	-
Screen color and pixel depth	-	-	-	-
Screen width and height	-	-	--	-
Screen availLeft, availTop, availHeight and availWidth	-	-	-	-
Screen hosizontalDPI, verticalDPI	-	-	-	-
Do not track	-	-	-	-
CPU	-	-	--	-
Language	High		High	High
Languages	-	-	-	-
Canvas element	-	-	-	--
Timezone	-	-	-	High

Notably, from the Table 5-8, the research discovered that changing User-agent both HTTP layer and JavaScript layer is highly problematic, more than other attributes. The research also established that using much-outdated versions of a web browser has a strong side effect on displaying of a web page, functionalities and difficulty to use, comparable to the latest version. Thus, this attribute cannot change arbitrarily without the understanding of its characteristics. Similarly, Language and HTTP header Accept-Language cannot change

arbitrarily because the web page will show the user with different languages that they could not understand in the content of a web page. For example, FP-Block changed Language attribute arbitrarily. Consequently, many websites used would show an abnormal web page. FP-Block had not given any reason for choosing particular attributes, likely selected arbitrarily. Furthermore, the research established that the rest of the attributes in Table 5-8 did not show strong side effects on a web browser.

5.3.2.3 The effects of information paradox.

The information paradox is one problem of current countermeasures that Eckersley (2010) mentioned in the course of collecting data of users on the Internet. He found that some web browsers gave some abnormal combinations to his database. For example, User-agent showed that device fingerprinting was an iPhone. It appeared that his algorithm could detect installing of Flash on the device, which is an abnormal combination. He mentioned that this made the web browser stand out rather than preventing the fingerprinting tracking. When the web browser's attributes are altered arbitrarily, the fingerprinting algorithm can easily notice and then find a new way to fingerprint the user computer. Therefore, this section would study whether each countermeasure possessed the abnormal combination or not.

In order to study problems of the information paradox, the web browser that installed each countermeasure would visit the hybrid fingerprinting website in order to observe whether the change of browser attributes created an abnormal combination or not. According to the experiment of Nikiforakis et al. (2013), they classified information paradox into three metrics: incomplete coverage of navigator object, inconsistencies between related attributes, and mismatch between HTTP header, details about metrics shown in Chapter 3.2. The result was shown in Table 5-9 as follows.

Table 5-9 showed the result of abnormal combinations resulting from producing each countermeasure. UserAgent Switcher for Chrome was extremely noticeable, showing inconsistencies between related attributes and mismatch between HTTP header. The Rubberglove was also obvious because it can be clearly seen that it ignored mismatch between HTTP header and incomplete coverage of navigator object. In addition, it showed only information about the user screen, but it is not showing any information related to the navigator object in JavaScript. FP-Block specifically changed loaded JavaScript objects

before displaying a web page. If any browser attributes were not run by a web page, FP-Block would not handle them. Handling only load attributes had created the abnormal combination more and more within the navigator object and inconsistencies between related attributes.

Table 5-9 Result of information paradox

Countermeasure	**incomplete coverage of navigator object**	**inconsistencies between related attributes**	**mismatch between HTTP header**
Rubberglove	✓	-	✓
UserAgent Switcher for Chrome	-	✓	✓
Canvas fingerprinting blocker	-	-	-
Fireglove	✓	-	-
FP-Block	✓	✓	
Stop fingerprinting	-	-	-

FP-Block was the first countermeasure designed to inhibit fingerprinting technique by considering consistency of attributes of the user computer. Unfortunately, the proposed FP-Block model remained relatively crude in concealing some attributes. As mentioned earlier, many suspicious attributes would be blocked by FP-Block before rendering a web page, while FP-Block would change only detected attributes, not all suspicious attributes. It meant that undetected attributes would be blocked by default. This problem still produced an information paradox similar to other fingerprint countermeasures.

There are many reasons why the remaining countermeasures were not showing abnormal combinations. Firstly, if the changed attribute was not associated with other attributes. Stop fingerprinting only altered font attribute. Usually, the font attribute was

not associated with other attributes. Therefore, this countermeasure did not produce any problems on the combination of attributes on the web browser. Similarly, Canvas fingerprinting blocker used canvas element to fingerprint the user computer. The canvas element has no relationship with other attributes on the navigator and screen object in JavaScript. Consequently, this countermeasure does not show an abnormal combination.

5.4 Research finding

This section is divided into two sections. Firstly, it briefly discusses the problems of existing countermeasures that are discovered during conduction of experiments. Secondly, it discusses the cause of problems by analysing the results of experiments.

5.4.1 **Problems of existing countermeasure**

Considering all experiments in this chapter, the existing fingerprint countermeasures had three problems: effectiveness of fingerprinting prevention, user browsing experience and information paradox.

Regarding the effectiveness of fingerprinting prevention, even though many fingerprint countermeasures showed the fingerprinting prevention with sufficient effectiveness, they cannot prevent four methods of fingerprinting technique. The one factor that was highly significant for preventing the browser fingerprinting technique was the number of changed attributes. If the number of changed attributes is sufficient, the fingerprint countermeasure would prevent more methods of fingerprinting technique than selecting fewer attributes. For example, Stop fingerprinting[63] randomized only fonts of the user computer. It seemed satisfactory to prevent the fingerprint tracking, but remaining attributes may be used to fingerprint if an attacker knew that some attributes are unchanged. In addition, considering the process of randomness and spoofing of attributes, many countermeasures using the technique of randomness and spoofing still produced the same fingerprinting ID. For instance, FP-Block[64] showed 93.3 percent of producing the different fingerprinting ID of object JavaScript fingerprinting, visiting the hybrid

[63] https://addons.mozilla.org/en-GB/firefox/addon/stop-fingerprinting/
[64] https://addons.mozilla.org/en-GB/firefox/addon/fp-block/

fingerprinting website for 30 times. Similarly, Stop fingerprinting showed 90 percent of producing the different fingerprinting ID for font fingerprinting.

It can be clearly seen that the technique to change the attributes is related to the user experience. The randomness and spoofing technique had insignificant side-effects on the user browsing experience, compared to the blocking technique. However, change of attributes with randomness and spoofing technique still had a problem of the user experience if a fingerprinting countermeasure did not consider which attributes could be changed or which attributes are unable to change. Furthermore, an increasing number of selected attributes are related to the user experience as well. If any fingerprint countermeasures chose more attributes, they definitely encountered more problems of user experience than choosing fewer attributes. Rubberglove was a good example of choosing excessively many attributes, namely 16 attributes. Thus, Rubberglove had more problems with the user browsing experience, compared to other countermeasures changing fewer attributes. In addition, the fingerprinting detection part made the browsing experience annoying to the user. The user felt stuck while browsing the web if a fingerprint countermeasure notified the user that they were being tracked and then forced the user to decide to prevent or not prevent the fingerprint tracking. For instance, FP-Block attempted to detect suspicious attributes and then inform the user to decide action. This made it extremely annoying to the user browsing the web.

In consideration of the information paradox, almost all tested fingerprint countermeasures still showed abnormal combinations (information paradox) after changing attributes. The incomplete coverage of navigator object, inconsistencies between related attributes and mismatch between HTTP header that caused the web browser to be easily distinguishable from a normal web browser still were not addressed.

5.4.2 **The causes of problem of existing fingerprint countermeasures**

Thoughtful consideration before changing the user identity is extremely important to improve effectiveness of fingerprinting prevention, to reduce problems with the user browsing experience and to diminish the information paradox. This section explains the problems of existing fingerprint countermeasures that are unsolved. Any problems of existing fingerprint countermeasures can be classified into five causes.

- **Ignoring some attributes**

Some fingerprint countermeasures ignored some attributes. The result of ignoring some attributes resulted in allowing the web browser to be tracked more easily, and making the web browser stand out. For example, Stop fingerprinting blocked plugins and randomized fonts. It seemed that this countermeasure can inhibit the fingerprint tracking, being capable of inhibiting generation of hybrid fingerprinting ID and font fingerprinting. However, the remaining attributes may be used to fingerprint, such as JavaScript object fingerprinting, as a result of ignoring JavaScript object.

- **Changing unsuitable attributes**

Some attributes cannot be changed arbitrarily. Changing them leads to problems of the user browsing experience and information paradox. For example, The User-agent Switcher for Chrome only changed the User-agent. Normally, the User-agent has a relationship with HTTP header and navigator object. Thus, this change caused the web browser to stand out because it cannot conceal some information related to the JavaScript object such as navigator object. In addition, changing User-agent with carelessness causes substantial side-effects on the web browser that means the user cannot use the web browser as usual.

- **Changing attributes without considering the relationship of attributes**

Many attributes are changed by the fingerprint countermeasure. They were suspected of being used to fingerprint. Changing attributes was relatively straightforward to prevent the fingerprint tracking, but the relationship of attributes should also be considered. Otherwise, the new combination of fingerprinting attributes may be an abnormal combination, easier to distinguish. Results of changing attributes with carelessness led to the information paradox and then made the web browser extremely noticeable, therefore easier to be fingerprinted. For example, FP-Block attempted to block 'Do not Track' in navigator object. It turned out that Do not Track still showed active on the HTTP header.

- **The number of changed attributes for preventing the fingerprint tracking**

The number of changed attributes directly related to the user experience, information paradox and effectiveness of fingerprinting prevention. The number of fingerprinting attributes needs to be a sufficient number. If the number of attributes was increased, it had a high risk of creating side-effects and generating an abnormal combination on the web browser. For example, Rubberglove handled more than 15 attributes. Thus, it was extremely difficult to avoid side-effects and creation of abnormal combinations on the web browser. On the other hand, if the fingerprint countermeasure selected few attributes, it appeared that the remaining attributes were at high risk of being fingerprinted. For example, Stop fingerprint selected only two attributes, list of fonts and plugins. It turned out that the remaining attributes were uncomplicated to fingerprint as this countermeasure did not handle any attributes except the list of fonts and list of plugins.

- **The used method for preventing the fingerprint tracking**

It cannot be denied that selecting a method to change attributes caused side-effects to the web browser. The blocking technique has a substantial impact on the web browser while randomness and spoofing technique was interesting to develop further. Rubberglove was a good example of using the blocking technique to prevent the fingerprint tracking. The result can be clearly seen that Rubberglove showed more problems of user browsing experience. While Stop fingerprinting or FP-Block used spoof and randomisation technique showed an insignificant impact on the web browser, this technique can be developed further in order to address the limitation of current fingerprint countermeasures.

5.5 Conclusion of this chapter

The main aim of this chapter was to find limitations of existing fingerprint countermeasures. In order to know the limitations of existing fingerprint countermeasures, the research had designed three main experiments: studying how the fingerprinting performs, validating the existing fingerprint countermeasures, and concluding limitations of each fingerprint countermeasure. The research commenced on the development of the hybrid fingerprinting website so that the results of a study of fingerprinting technique can be brought to create a defence of the fingerprinting technique in the future, and to examine

the effectiveness of fingerprinting prevention, side-effects and information paradox. The results of a preliminary experiment that revealed five problems of existing countermeasures still are unsolved: ignoring some attributes, changing unsuitable attributes, changing attributes without considering the relationship of attributes, the number of changed attributes for preventing the fingerprint tracking and the used method for preventing the fingerprint tracking.

Chapter 6 Proposed countermeasure

The results of preliminary experiments in the previous chapter showed considerable advantages in support of creating a new countermeasure to inhibit the fingerprint tracking with a minimum of side-effects to the web browser. In Section 6.1, this chapter starts by comparing the advantages and disadvantages of interesting countermeasures. Then, a new fingerprint countermeasure will be proposed and then steps in overall development will be explained in Section 6.3. Finally, Section 6.4 explains how to implement a new fingerprint countermeasure in practice.

6.1 Considering interesting countermeasures

This section selects interesting countermeasures to explore further. There are many countermeasures claiming that they provided a special method to reduce side effects caused by changing attributes on the web browser such as FP-Block (Torres et al., 2015). The research had also mentioned FPGuard (Weldemariam, 2015) and Privaricator (Nikiforakis et al., 2014b) which they are not tested in the preliminary experiment because they are not made available online. However, the results and their concepts are so interesting in terms of effectiveness of anti-fingerprint and still maintaining the normal level while users browse the Web. For example, Nikiforakis et al. (2014) maintain that their countermeasures show trivial side effects on a web browser. They present results of their countermeasure against the fingerprint tracking by observing fingerprint ID provided by fingerprinters and evaluate the user experience by observing a fraction of pixels when the website is loaded. Exploring the idea of these countermeasures will assist to create a new countermeasure to prevent the fingerprint tracking, brief details of each countermeasure described below.

6.1.1 Concept model of FP-Block

Torres et al. (2015) used the concept of web identities for preventing the fingerprint tracking. This countermeasure was called FP-Block. FP-Block had two requirements: created web identities must be sufficiently different from any previously created web identity and the created web identity must be consistent with the nature of the attributes.

The two requirements of FP-Block intended to resolve problems of fingerprinting prevention, user's browsing experience and information paradox.

To explain the concept of web identities, this concept believes that fingerprinting a user is not a substantial problem. On the other hand, bringing the user's fingerprint from one website to another is a real problem. It means that FP-Block did not pay attention to what kinds of script are run on the web browser. Instead, FP-Block will remember only the URLs of the websites visited and then generate fake attributes to each website that the user visits. It means that attributes will be changed only in line with a visited website. This concept regards that frequent change of attributes causes an increase of side effects on the web browser. On the other hand, infrequent change of attributes could balance usability and functionality.

For this reason, FP-Block will generate different web identities and use the generated different web identities to break the linkability as shown in Figure 6-1.

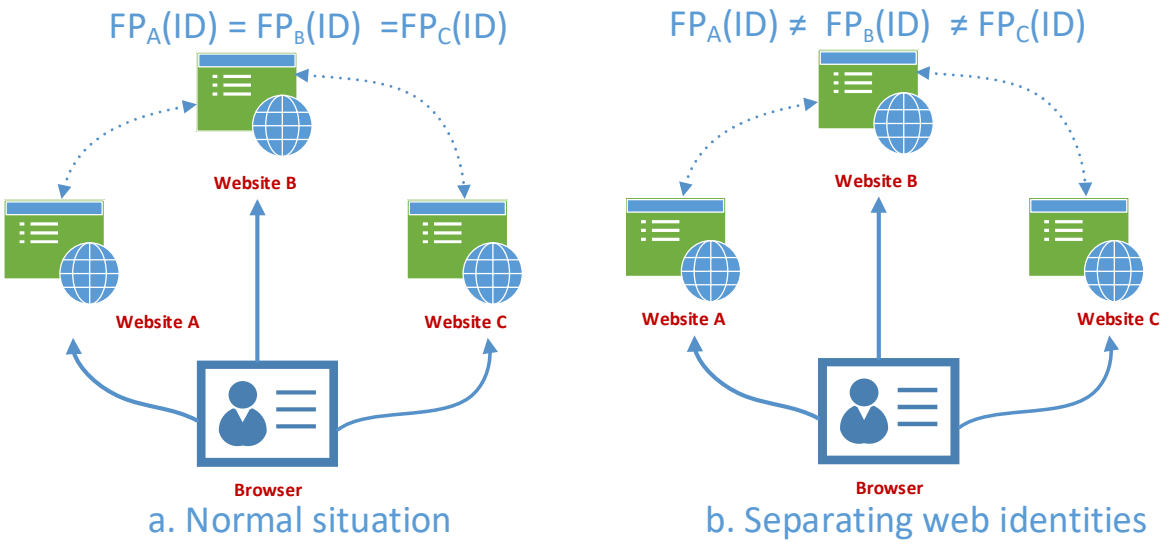


Figure 6-1 concept of different web identities of FP-Block

In order to create different web identities, 22 attributes had been selected. Torres et al. (2015) had grouped attributes as shown in Table 6-1. They had classified changed attributes into five groups, namely browser, language, OS/CPU, screen and Timezone. The real concept of FP-block is to change 22 attributes as though they are genuine attributes. With this concept, the attributes will be changed only once because they regard that

changing attributes many times may increase the problems of user experience, especially functionalities of a web browser.

Table 6-1 groups of attributes selecting for creating distinct web identities

Type of attribute	**Name of attribute**
Browser	User-agent, Browser name, vendor, accept-encoding
Language	Language, user language, system language
OS/CPU	CPU class, oscpu, platform
Screen	Width, Height
Timezone	Time zone offset

Note that FP-Block was the only fingerprint countermeasure that considered the problem of information paradox. FP-Block attempted to address the abnormal combination of attributes within JavaScript layer and HTTP header fields by applying a generic model of Markov chain as shown in Figure 6-2 and Figure 6-3. Markov chain will be served as a distribution of attributes to a web browser, while the probability of transition of each attribute is derived from the code of J. Mealo's data[65]. This model would change each attribute in line with the model of FP-Block that randomizes attributes one by one, based on the real relationship of attributes.

[65] https://github.com/jmealo/random-ua.js

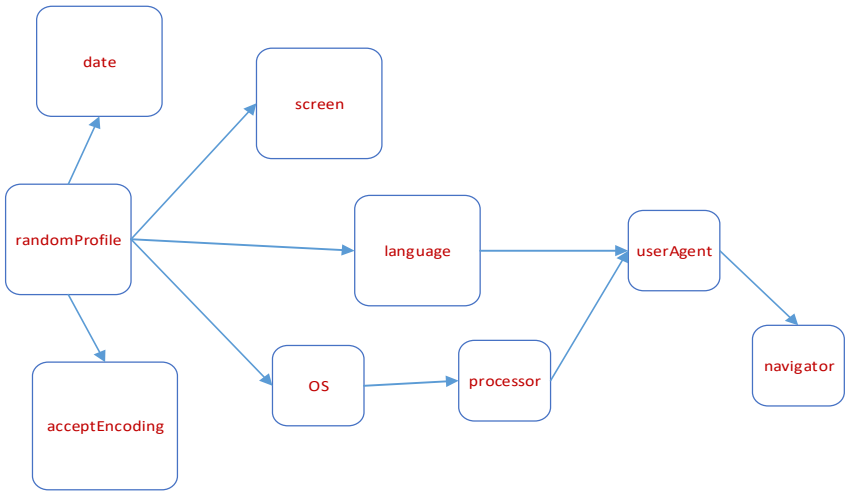


Figure 6-2 the model of FP-Block

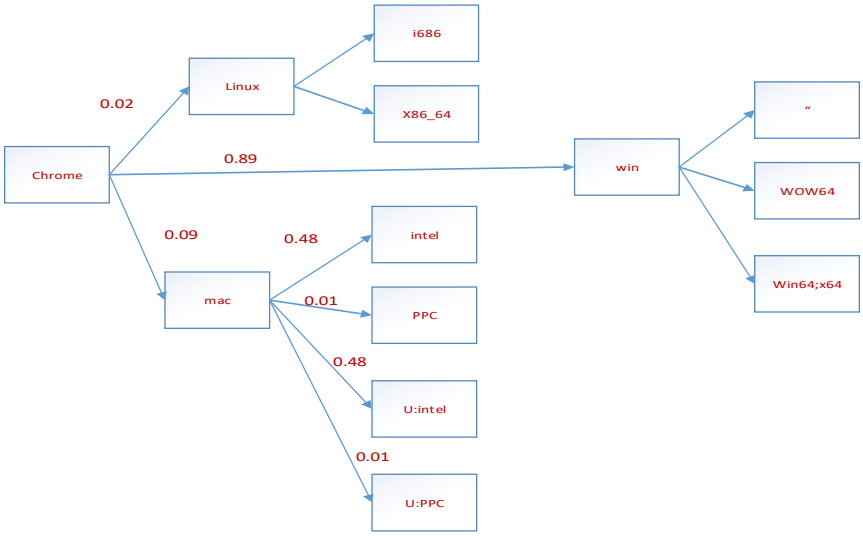


Figure 6-3 partial Markov chain used to create web identities for Chrome profile

Unfortunately, this concept did not provide any concrete results that it could maintain a perfect balance between the usability and functionalities. In addition FP-Block could prevent only the method of object fingerprinting (results from previous chapter), and was unable to cover methods of fingerprinting technique based on JavaScript. In terms of mitigation of problems of information paradox, FP-Block is unable to create a normal combination due of lack of understanding of characteristics of attributes. Randomizing attributes one by one based on the real relationship seems to be a promising idea. In reality, applying a Markov chain model to distribute attributes becomes useless because each

randomized attribute did not show a possible combination in practice. For example, the model of FP-Block attempts to randomize operating system, processor and browser version in line with the randomized User-agent. It turned out to be that randomized attributes do not match with User-agent used in reality. Lack of understanding of sequence of changing attributes means that the problem of information paradox is still unsolved. Furthermore, unawareness of characteristics of attributes is responsible for increasing side effects on the web browser. For instance, FP-Block sometimes changes '*Language*'. Changing attributes in such a way brings about incorrect display.

6.1.2 Concept of Privaricator

Nikiforakis et al. (2014b) propose to deal with the problem of side effects by changing only a few attributes. Their countermeasure is called Privaricator. They accept the idea of changing high entropy attributes that is satisfactory to prevent tracking. Instead of blatant lying, they employ randomization policies for changing attributes. The level of changed attributes depends on the detection part detecting whether the loading of attributes is abnormal or normal. If suspicious attributes load abnormally, Privaricator will change suspicious attributes in line with their policies. They offer an interesting idea that a web browser is usually updated automatically once a week. Thus, selecting attributes unrelated to infrequent updates of a web browser could extend their countermeasure to be used over time. In addition, they pointed out that the change of User-agent is more disadvantageous than advantageous because User-agent is associated with many attributes of both HTTP layer and JavaScript layer.

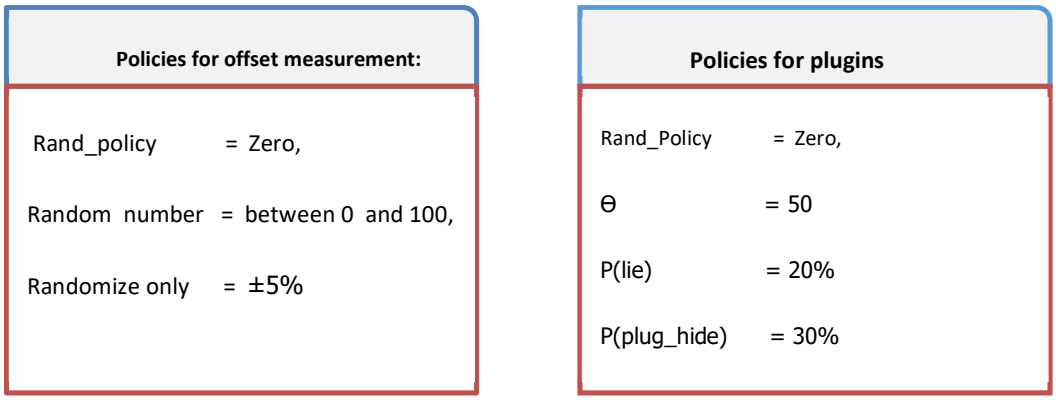


Figure 6-4 policies of Privaricator

As mentioned earlier, they used randomization policies in order to reduce side effects on a web browser. The first policy involved offsetWidth, offsetHeight and getBoundingClientRect. These attributes are related to the loading of fonts. The second policy involved loading of plugins, details of policies shown in Figure 6-4.

When the policies of offset measurement are operative, rather than returning the original offset value, they will return zero, a random number from 0 to 100 and the original offset value ±5% to the web server. For randomization of plugins, Privaricator will hide the plugins list of a browser at 30% if navigator.plugins is loaded and lie about plugins at 20% after 50 offset accesses.

Changing attributes in such a way, Privaricator believed that it could mitigate problems of user browsing experience better than other countermeasures. However, they did not show any reasons why they defined policies in this fashion. In addition, disregarding to change User-agent seems to be a time bomb because the result from Table 2-2 clearly indicates that many fingerprinters always employ User-agent to fingerprint.

6.1.3 **Concept of FPGuard**

Weldemariam (2015) had developed FPGuard with the concept of changing attributes based on the detection part, similar to Privaricator (Nikiforakis et al., 2014b). The difference between FPGuard and Privaricator is that FPGuard does not use policies to change attributes. Instead, FPGuard monitors indicators of the fingerprinting efforts. In order to reduce the arbitrary randomness, if the access of suspicious attributes exceeds the threshold, FPguard will change detected attributes in line with its detection algorithm. FPGuard defined nine metrics as indicators for fingerprinting attempts, details shown in Table 6-2.

FPGuard regarded that fingerprinting efforts should be separated into nine metrics to be able to mitigate problems of user browsing experience rather than arbitrary change. It means that FPGuard will only change attributes according to detected attributes. However, changing attributes without regard to the relationship of other attributes contributes to the problem of a web browser standing out instead. Furthermore, FPGuard admitted that relying only on the detection part might lead to incorrect prevention because

of complex and hidden fingerprinting script by fingerprinters that are difficult to detect accurately (FaizKhademi et al., 2015).

Table 6-2 metrics defined as indicators for fingerprinting efforts

Metric Description	**Detection**	**Prevention**
Metric 1	The number of accesses to the navigator and screen objects'	Randomness
Metric 2	The number of accesses to the properties of the Plugin and mimeTypes	Randomizing virtual plugins
Metric 3	The number of fonts loaded using JavaScript	Randomizing fonts
Metric 4	The number of accesses to the offset properties of HTML	Adding noise
Metric 5	Checks whether a canvas element is programmatically accessed	Adding noise to canvas element
Metric 6	Checks visibility status (hidden or visible) of a canvas element that is programmatically accessed	Adding noise to canvas element
Metric 7	Checks the existence of methods for enumerating system fonts	Randomizing fonts
Metric 8	Checks the existence of methods for transferring the collected information	Adding noise
Metric 9	Checks the visibility status of Flash	Disable Flash

6.1.4 Comparing and contrasting interesting countermeasures

Regarding three fingerprint countermeasures that focus on problems of user experience, there are a number of similarities and dissimilarities in some respects between the three countermeasures (FP-Block, Privaricator and FPGuard).

With respect to the concept of the three countermeasures, different methods and a different number of attributes have been selected to tackle problems of user browsing experience. Both Privaricator and FPGuard assert that effective detection could assist the prevention part to result in more effective prevention with the minimum impact to the web browser. Unlike the previous two countermeasures, FP-Block selects to use a different concept: FP-Block offers the idea of web identities to break the linkability and maintain functionalities of a web browser as usual. Using this concept, FP-Block does not rely on the detection part, details shown in Table 6-3.

In terms of a number of selected attributes, it can be clearly seen that the three countermeasures handle suspicious attributes with different concepts. Some attributes had been selected based on the idea that changing these selected attributes is sufficient to break fingerprinting. Privaricator proposes to choose few attributes because changing more attributes is correlated with more side effects on the web browser. FPGuard and FP-Block desire to cover remaining attributes that may be used to fingerprint. Thus, choosing more attributes is inevitable for them.

From the research findings in the previous Chapter, many fingerprint countermeasures handle insufficient attributes to prevent four methods of a browser fingerprinting technique, ignore problems of information paradox, disregard problems of user browsing experience and overlook the consequences of changing suspicious attributes. As in Table 6-3, only Privaricator pays attention to the problem of changing attributes while FP-block is the only countermeasure that is attentive to solving the problem of information paradox.

To sum up, three interesting countermeasures use different concepts to mitigate the problems of user browsing experiences and use different attributes to prevent the fingerprint tracking. The fingerprinting detection part is not always necessary for preventing the fingerprint tracking with minimum impact to the web browser. In addition, none of any countermeasures can cover four vulnerabilities, namely covering noticeable

attributes, mitigating problems of information paradox, mitigating problems of user experiences and considering a consequence of changing attributes.

Table 6-3 compare and contrast of each countermeasure

Countermeasures	**Method to mitigate the problem of user browsing experiences**	**A number of handled attributes**	**Using detection part**	**Using prevention part**	**Method to change**
FP-Block (Torres et al., 2015)	Using concept of web identities	19 attributes	-	✓	Spoofing
Privaricator (Nikiforakis et al., 2014b)	Using Randomization policies	4 attributes	✓	✓	Randomization
FPGuard (Weldemariam, 2015)	Using changing attributes based on metrics.	32 attributes	✓	✓	Blocking and randomization

Table 6-4 the differences of each fingerprinting countermeasure

Countermeasures	**Covering noticeable attributes**	**Mitigating problems of information paradox**	**Mitigating problems of user browsing experiences**	**Considering consequence of changing attributes**
FP-Block (Torres et al., 2015)	✓	✓	✓	-
Privaricator (Nikiforakis et al., 2014b)	-	-	✓	✓
FPGuard (Weldemariam, 2015)	✓	-	✓	-

6.2 Proposing a new countermeasure

With respect to Table 6-3, and 6-4, the three countermeasures are still vulnerable to attack from the browser fingerprinting technique. In order to fix limitations of the three countermeasures, this section proposes a new countermeasure called **FP-prevention.** The in-depth understanding of how the fingerprinting technique works and what are existing vulnerabilities from preliminary experiments in Chapter 5, was a key heart of the design of FP-prevention. FP-prevention will handle existing vulnerabilities of the three countermeasures.

6.2.1 General features of a new countermeasure

A fingerprint countermeasure is a solution designed to tackle the problem of the browser fingerprinting technique. With the intention of preventing unwanted tracking, it should have the following features.

- **The proposed countermeasure should handle most main methods of JavaScript-based tracking.** This research did not accept the idea that changing few attributes such as Privaricator (Nikiforakis et al., 2014b) was the best method of fingerprinting prevention. Therefore, in order to prevent remaining attributes being fingerprinted, the number of changed attributes should be appropriate in order to disengage tracking. To consider the aim of a number of methods of fingerprinting technique inhibited by a new countermeasure, the research would focus only on popular attributes used by fingerprinters: JavaScript object fingerprinting, font fingerprinting, plugin fingerprinting and canvas fingerprinting.
- **The proposed countermeasure should not create side effects on the web browser.** Results from the preliminary experiment in Chapter 5 indicated that changing attributes with the wrong method, handling attributes with unawareness of the characteristics of changed attributes, finally results in side effects on the web browser. Many fingerprint countermeasures modify attributes arbitrarily and do not regard consequences of changing attributes. The incorrect modification could degrade the user browsing experience until a user is unable to browse a web page as usual. The process of changing attributes should be a key factor to prevent problems with the user experience, reducing unwanted side-effects.

- **The proposed countermeasures should not create abnormal combinations.** Regarding attributes in Table 2-1, both fingerprinters and fingerprint countermeasures mainly focused on the high entropy attributes, while value of HTTP header fields were ignored by almost all fingerprint countermeasures in spite of being the main cause of information paradox. Many attributes, such as operating system, web browser name and web browser version, were changed by many fingerprint countermeasures without considering their relationship. The proposed countermeasure will bridge the gap between HTTP layer and JavaScript layer with the purpose of addressing problems of inconsistency between related attributes.
- **The proposed countermeasure should perform automatically without detection part.**

  Mostly, many fingerprint countermeasures were designed to detect the fingerprinting code, such as FPGuard (Weldemariam, 2015) and Privaricator (Nikiforakis et al., 2014b). Typically the main idea of detection part was to detect the loading of suspicious attributes and then find a proper method of handling each method of fingerprinting technique. It turned out that many countermeasures did not show concrete results that the detection part would assist fingerprinting prevention better than without the detection part. For example, FPGuard detects loading of suspicious attributes while the web browser is loading a web page and then applies an appropriate method to prevent tracking of each method of fingerprinting technique before displaying a web page. However, there were no relevant results of an experiment of FPGuard that using detection part would perform more effectively than not using it. According to Orr et al. (2012) using static analysis to detect and block JavaScript, Tran et al. (2012) using dynamic analysis and Acar et al. (2013) using behavioural analysis to detect fingerprinting script, they all concluded that it was extremely difficult to detect the JavaScript code accurately. Therefore, the inaccurate detection of JavaScript code probably produced negative side-effects on the web browser rather than supporting the fingerprinting prevention. Another example is the concept of web identities, FP-Block (Torres et al., 2015) , seemed to be good but in practice, the user needed to click '*Keep blocking*' for FP-Block to remember URLs of websites. This made users annoyed while navigating websites. The proposed countermeasure will do the opposite. It

will not ask any decision from the users while navigating. This would mitigate the problems of having many notifications on the web page.

6.2.2 Intuition behind the proposed countermeasure.

The main source of the browser fingerprinting technique derives from a diversity of attributes on the user computer. Conversely, the diversity can be reversed as a strategy to prevent the fingerprint tracking. Even though the previous three countermeasures attempt to use this strategy, their ideas are still not complete to cope with problems arising from the fingerprinting prevention. Therefore, the proposed countermeasure will propose a different idea from the three countermeasures in order to fix existing problems, namely inability to cover an important method of the browser fingerprinting technique, side effects on the web browser, and information paradox. Typically, the process of most countermeasures development is based on two phases, as shown in Figure 6-5.

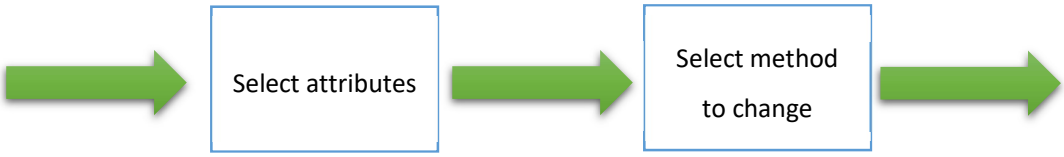


Figure 6-5 traditional process of developing countermeasures

The process of development in traditional fashion (Figure 6-5) has resulted in problems with a web browser as previously mentioned. It can be clearly seen that the traditional process causes problems of fingerprinting prevention. Hence, the proposed countermeasure will employ a new process based on in-depth understanding of how fingerprinting techniques work, how existing countermeasures prevent tracking, and results from Chapter 5, to solve existing problems of fingerprinting prevention. The new process is shown in Figure 6-6.

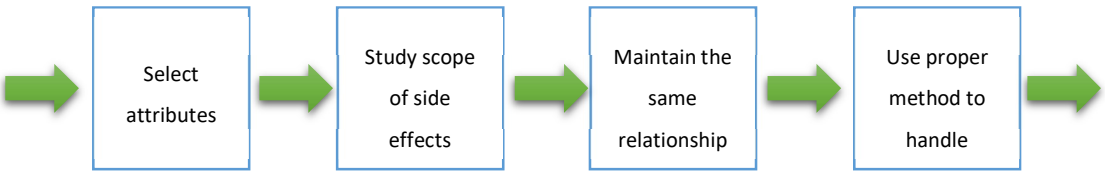


Figure 6-6 a new process of developing proposed countermeasure

The final outcome of the process in Figure 6-6 will be constructed into FP-prevention model. All phases will be clarified further in the next section. This created model is powerful enough to handle existing problems of fingerprinting prevention.

6.3 Steps in development of FP-prevention

The key idea of FP-prevention was the approach of misreporting parts of user environments at runtime. Definitely, this research required to misreport user environments which would break tracking with the minimum side-effects on the web browser. As mentioned in Section 6.2, the research does not see the need for detection part because it is difficult to detect the fingerprint script accurately (Tran et al., 2012, Acar et al., 2013, Cova et al., 2010). By contrast, changing attributes on the user computer is sufficient to prevent the fingerprint tracking. The question has been raised how to change attributes with the minimum side effects on a web browser, and to have changed attributes not so noticeable that a web browser stands out.

FP-prevention introduces the concept of the sequence of correct development to be able to assist the fingerprinting prevention, mitigate problems of user experience and reduce the information paradox. With this idea, the research had developed FP-prevention model to address these problems. From the introductory results in Chapter 5, the research accepted that the design of proposed countermeasure should consist of four phases as follows.

Phase 1: Selecting attributes

The first phase is an essential part of the decision as to what attributes should be employed for obfuscating the process of identifying a user with the browser fingerprinting technique. With the terrific number of user attributes, countermeasures cannot select all attributes that are suspicious. For this reason, the number of selected attributes that is sufficient will play a crucial role instead. Considering results from preventing methods in Table 5-4, attributes that are subject to fingerprinting should be covered. Thus, the first step of the first phase should be to choose attributes that are suspicious, and selected attributes should cover popular methods of the browser fingerprinting technique. Next, the second step is to decide which attributes could compromise the user experience and

then take problematic attributes away from the list. Selecting attributes in such a way, the research needs to know whether selected attributes have more or fewer side effects on the web browser. For the second step, the decision of which problematic attributes to ignore had been made by utilizing introductory results from Table 5-7 and Table 5-8 for consideration. Following the two steps in this phase can mitigate the problems of uncovering suspicious attributes to be fingerprinted and side effects on the web browser. According to Table A-1 in Appendix A, attributes can be classified into two types.

1. Flexible Attributes

These attributes can be changed without any significant side-effects to the web browser such as DNT, online or product-Sub. However, it does not mean that the changed attribute will not show any side-effects at all.

2. Inflexible attributes

These attributes are unable to be controlled because interception may make the web browser stand out more and develop side-effects. For example, if any countermeasures changed the Language, the web browser will display the wrong language which a user is unable to understand.

Phase 2: Knowing the scope of side effects when changing attributes

Despite having suitable attributes for randomization, this is not sufficient to prevent side effects on the web browser. From the preliminary experiment in the last chapter, some attributes did not show any side effects at first, but they gradually showed side effects more and more when the used sample is changed. This cause derives from ignorance about the characteristics of attributes. For instance, a sample of last User-agent may not show any side effects while browsing a web. By contrast, if randomizing User-agent leads to using an outdated version of User-agent, the web browser will show significant side effects to the web browser because many websites attempt to customize their web page to be up-to-date with the latest version of a web browser. Thus, providing a suitable data is essential for fingerprint countermeasure. Any data that seems to have problems with a web browser should be used with caution.

From the reason above, it can be clearly seen that collecting a data for randomization is not a substantial problem when compared with using an incorrect data

with the web browser. This is the main cause that many countermeasures are still impractical.

Phase 3: Maintaining the same relationship

After going through two steps, the selected attributes would be considered by their relationship in the FP-prevention model. The relationship of the created model will assist to create a combination of attributes that are similar to a real browser. The gap between information on HTTP header files and JavaScript caused by an impossible combination would be bridged by FP-prevention model. The FP-prevention model considers the most problematic attribute (User-agent) to handle it first because it is associated with HTTP layer and JavaScript layer. Then, attributes that are associated with User-agent will be considered for the next relationship until FP-prevention model is constructed as shown in Figure 6-7.

Handling problematic attributes is the key idea to tackle the problem of information paradox and user experience, rather than ignoring them. For example, regarding change of User-agent, many published papers understand that it makes more problems of user experience rather than reducing the problem such as Broenink (2012a) and Nikiforakis et al. (2014b). The concept of FP-prevention thinks differently from many researchers. The problem of User-agent should be fixed instead of ignored because User-agent is extremely likely to be used for fingerprinting as detailed in Table 2-2. The selected attributes will be arranged as in FP-prevention model and are then used for reducing problems of information paradox.

This section lists the attributes used by fingerprinters and fingerprint countermeasures, shown in Table 6-5. Obviously, fingerprinters will choose attributes that they think are suitable for them. On the other hand, the fingerprint countermeasure needs to guess which attributes are sufficient to prevent the fingerprint tracking with the minimum impact on the web browser. This is a difficult decision when developing the fingerprint countermeasure.

Considering Table 6-5, there are about 41 attributes used for fingerprinting. It seemed a large quantity, but it was extremely helpful in showing why fingerprint

countermeasures prefer to choose attributes in such a way. Because of the diversity of attributes on the user computer, some attributes might be manipulated, while some attributes might not be manageable such as Date & Time and Timezone. If these problematic attributes are handled, unwanted side-effects and information paradox will be made. It is evident that difference of attributes makes it difficult for a countermeasure to handle them with minor effects.

Comparing which countermeasures can handle more attributes than another countermeasure is not correct. The number of handled attributes should not be compared. In contrast, observing the results of interception that handled attributes that are able to prevent methods of fingerprint tracking, that show insignificant side effects, and that show negligible problems of information paradox, should be a correct method.

A number of selected attributes for FP-prevention in Table 6-5 had considered the relationship with other attributes, side effects of change, and covers popular methods of browser fingerprinting technique based on JavaScript. These selected attributes are regarded sufficient to tackle problems of fingerprinting prevention.

- **Unselected attributes**

In order to balance usability and side-effects, some attributes used for fingerprinting would not be handled because handling these attributes does not help in maintaining the consistency of attributes and could produce a high risk of unwanted side-effects to the web browser as shown in Figure 6-7.

Table 6-5 comparison of used attributes by fingerprinters and countermeasures

	Fingerprinters							Countermeasures			
Attribute	**Pan**	**BC**	**IO**	**TM**	**Add**	**FPjs**		**FPP**	**PV**	**FPG**	**FPB**
List of plugins	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓
List of fonts	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓
User-Agent	✓	✓	✓	✓	✓	✓		✓		✓	✓
HTTP Header Accept	✓							✓			
HTTP Header Accept-Charset	✓										
HTTP Header Accept-Encoding	✓										✓
HTTP Header Accept-Language	✓										✓
Screen Resolution	✓	✓	✓	✓	✓	✓		✓		✓	✓
Timezone	✓	✓	✓	✓	✓	✓					✓
Browser language	✓	✓	✓	✓	✓	✓		✓		✓	✓
Operating system		✓	✓	✓	✓	✓		✓		✓	✓
DOM Storage	✓	✓	✓	✓	✓	✓					✓
IE userdata	✓	✓									
Java Enabled	✓				✓	✓		✓			✓
DNT User Choice					✓	✓				✓	✓
Cookies Enabled	✓		✓					✓		✓	✓
JS detect: Flash Enabled	✓	✓	✓	✓	✓	✓		✓		✓	✓
ActiveX + CLSIDs	✓	✓	✓	✓	✓	✓					
Date & Time		✓	✓	✓	✓						
CPU		✓	✓		✓	✓				✓	✓
System / User Language	✓	✓	✓		✓	✓				✓	✓
openDatabase			✓		✓	✓					✓
Canvas Fingerprinting					✓	✓		✓		✓	✓
Mime-type Enumeration	✓			✓				✓		✓	✓
HTTP Proxy Detection			✓	✓							
IndexedDB					✓	✓					✓
Math Constants		✓			✓						
Windows registry		✓	✓								
TCP/IP Parameters		✓	✓								
Google Gears Detection		✓									
Flash Manufacturer				✓							
MSIE Security Policy		✓									
AJAX Implementation		✓									
MSIE Product key			✓								
Device Enumeration		✓	✓								
Device Identifiers			✓								
IP Address		✓									
HTML Body Behaviour						✓					
Battery					✓						✓
WEBGLRenderingContext					✓						✓

Fingerprinters	Cited from	FingerPrint Countermeasure	Cited from
Pan	Panopticlick	FPP	FP-prevention
BC	BlueCava	PV	Privaricator
IO	Iovation	FPG	FPGuard
TM	ThreatMetrix	FPB	FP-Block
Add	AddThis		
FPjs	FingerPrintJS		

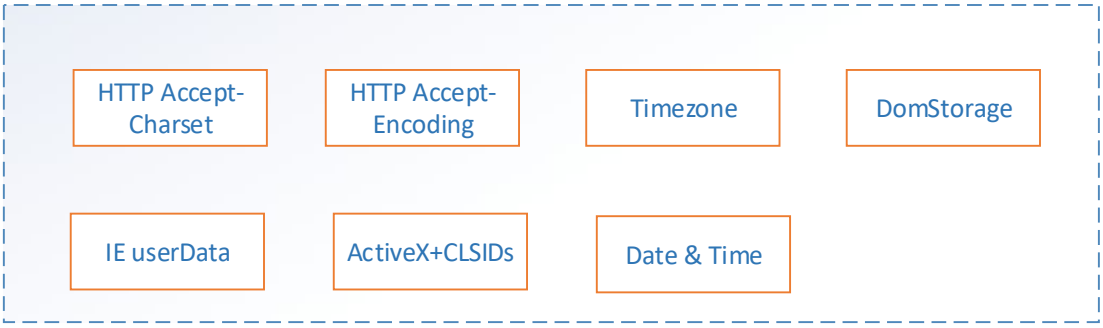


Figure 6-7 unselected attributes to prevent fingerprinting

In fact, there are many request header fields (about 35 request fields) used for communication between client and server. With regard to Table 6-5, there are only four request fields used for fingerprinting. FP-prevention had selected only User-agent on HTTP header field for assembling with other randomized attributes which the research had regarded is sufficient to produce diversity of fingerprinting ID and to maintain the consistency of attributes. Accept-Language, Accept-Charset and Content-Encoding on HTTP header should not be handled because HTTP Accept charset involves character set, unrelated to a created combination, while HTTP Content-Encoding is involved in encoding content. FP-prevention would handle attributes in HTTP header fields as necessary.

Timezone and Date&time have a relationship together. These attributes are often used to check the user's location. Therefore, handling these attributes presents a high risk to user browsing experience and user login.

DomStorage and IE userData are low entropy attributes. Interfering with these attributes is not worthwhile. Therefore, FP-prevention would safely ignore these attributes.

ActiveX and CLSIDs is a product of Microsoft technology. These technologies mainly involve Microsoft products. Handling this attribute without sufficient understanding leads to a web browser standing out because there are many ways to check that these attributes are installed on the user computer.

The remaining attributes are not mentioned because these attributes did not show concrete results (entropy value) of sufficient accuracy to be used in practice.

Attributes that are subject to fingerprinting will be selected to use with FP-prevention. Prior to the consideration of selected attributes taking place, FP-prevention must go through the previous two processes as explained earlier. The selected attributes are mainly associated with the Object fingerprinting, font fingerprinting, plugin fingerprinting and canvas fingerprinting.

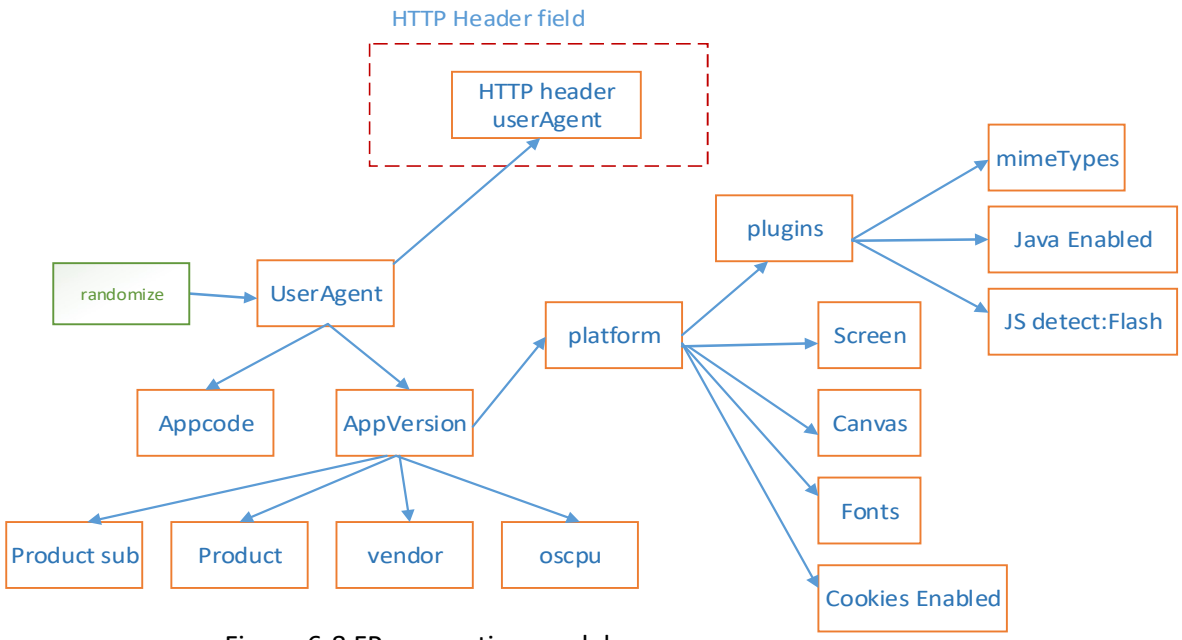


Figure 6-8 FP-prevention model

Phase 4: Using a proper method to handle selected attributes

Given the result from Table 5-6 and Table 5-7, it showed that the blocking technique has substantial impacts on the web browser. In order to avoid this problem, FP-prevention will avoid the blocking technique and uses the randomization instead. The randomization technique is selected because introductory results from the previous chapter in Table 5-6, Table 5-7 and Table 5-8 indicated that using randomization technique did not degrade the user experience, compared to the blocking technique. The research concurred with the idea of Nikiforakis et al. (2014b) that the real offender behind the browser fingerprinting technique is not the fact that a fingerprinted user is unique, but it is the linkability – the capability to link the user with the same fingerprinting ID. Tor[66] altered many attributes on the web browser in order to make all the Tor users look the same. Instead of using the concept of Tor, the notion of FP-prevention was the opposite.

Notably, the randomization technique was not simple as it can cause side-effects resulting from an impossible combination (Nikiforakis et al., 2013) as well as blatant lying being not a good notion. The sequence of randomized attributes will follow the FP-prevention model, so that randomized attributes still sustain the same relationship. This can mitigate the problem of information paradox.

FP-prevention alters attributes every time a user visits websites by using the randomization technique based on FP-prevention model. The changed attributes would annihilate linkability from one website to another, thus preventing the fingerprint tracking.

6.4 Implementation

FP-prevention would be implemented on Chrome web browser[67] as the number of users with Chrome web browser is more substantial than other browsers[68] and many APIs and documents are freely and widely available on the Internet. FP-prevention was able to work in private browsing mode. The integration of countermeasures of stateful tracking

[66] https://www.torproject.org/download/download-easy.html.en
[67] https://developer.chrome.com/extensions
[68] https://www.w3counter.com/globalstats.php

and stateless tracking would assist the user privacy in terms of reducing the probability of being fingerprinted (reducing fingerprinting surface[69]).

6.4.1 Operation of FP-prevention

The diversity of attributes customized by a user is a good ingredient of the browser fingerprinting technique. On the other hand, it can be used to counter the fingerprint tracking. When this idea is applied with FP-prevention, it is powerful enough to tackle existing problems of fingerprinting prevention. The idea of FP-prevention will randomly select a coherent set of attributes (e.g., User-agent, plugins and mimeType). The large set of attributes assists FP-prevention to display different fingerprinting ID. For this reason, FP-prevention can annihilate one important characteristic (stability) of exploitation by using the browser fingerprinting technique. The problem of a web browser standing out will be mitigated because the randomized attributes still sustain the same relationship. With selecting suitable attributes, problems of user browsing experience will decline because selected attributes are handled properly.

Given Figure 6-7, the workflow of FP-prevention model would start with the randomness of the User-agent. The value of this attribute was randomly selected first because of its association with the most side effects of the web browser and because it has a relationship with many attributes of both HTTP header file and JavaScript object. Normally, the User-agent contains much information inside as described in Table 6-5.

Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.94 Safari/537.36

[69] https://www.w3.org/TR/fingerprinting-guidance/

Table 6-6 information inside of the User-agent

Chrome 50.0.2661.94	
Mozilla/5.0	Mozilla version
Windows NT 6.1	Operating system (Windows 7)
WOW64	A 32-bit application is executing on a 64-bit processor
ApplewebKit	a set of core classes to show web content in Windows
537.39	Web Kit build
KHTML	Open source HTML layout engine developed by KDE project
Gecko	Web browser engine developed by Mozilla Foundation
Chrome	Chrome
50.0.2661.94	Chrome version
Safari	Based on Safari
537.36	Safari build

FP-prevention had collected 100 samples of User-agent string from the site[70], consisting of Mozilla Firefox 38+, Google Chrome 37+, Internet Explorer 11+ and Opera 45+. As soon as a web browser sends a request to a web server, User-agent will be randomized by FP-prevention.

Randomizing User-agent at once can obtain much information about the web browser (e.g., name, version and engine of the web browser) and operating system (name of operating system and number of bits CPU). As soon as User-agent is randomized, it will be copied to the User-agent on the HTTP header file and User-agent in the JavaScript object

[70] https://developers.whatismybrowser.com/

(navigator.userAgent). Then, the randomized User-agent would split the data into navigator.appCode, and navigator.appVersion as shown in Figure 6-8 below.

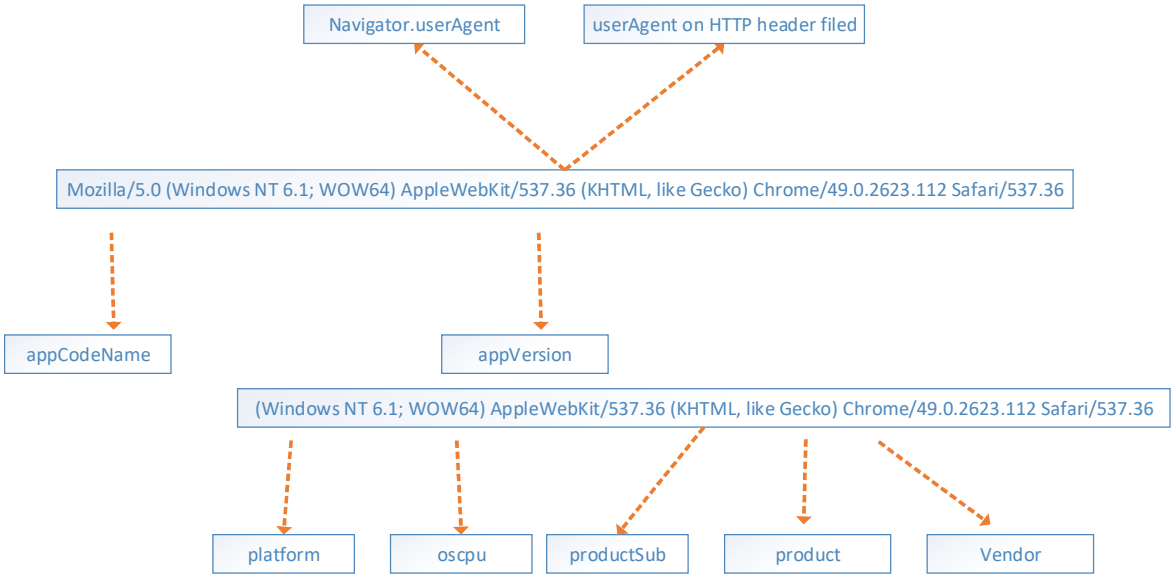


Figure 6-9 workflow of splitting User-agent string

Considering navigator.appVersion, it would convert into a platform of the web browser (navigator.platform), the current operating system (navigator.oscpu), the product name of the current browser (navigator.product), the name of the browser vendor (navigator.vendor) and the build number of the current browser (navigator.productSub) respectively, more details of randomization showing in Appendix I.

Let's say that:

AppVersion = (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.94 Safari/537.36

It means that this is a Chrome Browser using Gecko (browser's engine), running on Windows 7 (64-bit) with version number 6.1 and using Chrome version *50.0.2661.94.*

As for the operating system, the obtained appVersion contains the name of the operating system (navigator.platform) used by the user. FP-prevention would randomize a list of plugins corresponding to the obtained name of the operating system. For example, if the obtained platform is Windows, FP-prevention model will randomize plugins in line

with Windows. Similarly, randomized mimeType must be consistent with randomised plugins, details of the relationship between mimeType and plugin as described in Figure 5-1.

With respect to attributes of screen, consisting of *height (screen.height), width (screen.width), ColorDepth (screen.colorDepth), and pixelDepth (screen.pixdelDepth),* FP-prevention will randomly select these attributes. From the statistic[71], there are popular desktop screen resolutions that users favour, namely: 1366x768, 1920x1080, 1280x1024, 1024x768, 1280x800, 1440x900, 1680x1050, 1280x720, 1360x768 and 1280x768. The colour depth would be randomized to the range from 24 to 32 bits.

The font attribute is an independent attribute that does not relate to other attributes. FP-prevention would randomise it to the range from 10 to 460 fonts for preventing font fingerprinting. The list of fonts derive from fingerprintjs library[72], around 460 fonts - Windows: 231 fonts, Mac OS: 167 fonts, and other operating systems: 62 fonts.

FP-prevention would add noise to the canvas element before rendering the webpage. This is sufficient to inhibit canvas fingerprinting.

Three operating systems, Windows (consisting of Windows 10, Windows 8.1, Windows 8.0 and Windows 7), Linux and Mac, will be used to randomize in line with the User-agent.

6.4.2 **Modify HTTP Header**

The Hypertext Transfer Protocol (HTTP) is a method for transporting information between a web server and a client. Every web browser and website need to follow the HTTP protocol. With more information sent to the HTTP header field, it can easily reveal the personal information of a user computer such as browser name, browser version, operating system, CPU. The HTTP header is regarded as providing sufficient information for creating the fingerprint tracker because it is sent with every request during interaction with a web browser (Yen et al., 2012b).

[71] https://www.w3schools.com/browsers/browsers_display.asp
[72] https://github.com/Valve/fingerprintjs2

From Table 6-4, it can be seen that FPGuard failed to address the problem of information paradox. Changing particular attributes on the JavaScript layer without considering the relationship of changed attributes with information on HTTP header fields is a considerable mistake, leading to problems of information paradox. In order to address this problem, the consistency between the JavaScript object (navigator) and HTTP header fields must be matched. Usually, the communication from a web browser to a website is done through HTTP protocol. The web browser must send a HTTP request to a website and then the website will acknowledge with HTTP response as detailed in Figure 6-9 below.

Figure 6-10 the information between request header and response header

Chrome.webRequest is one API that does the function of intercepting and modifying HTTP requests. The Chrome extension had defined the life cycle of a web request and set of events, as shown in Figure 6-10. Thus, mitigation of abnormal combination between information on JavaScript layer and HTTP layer can be implemented by using the available API on the Chrome extension to modify information on HTTP header fields in HTTP layer.

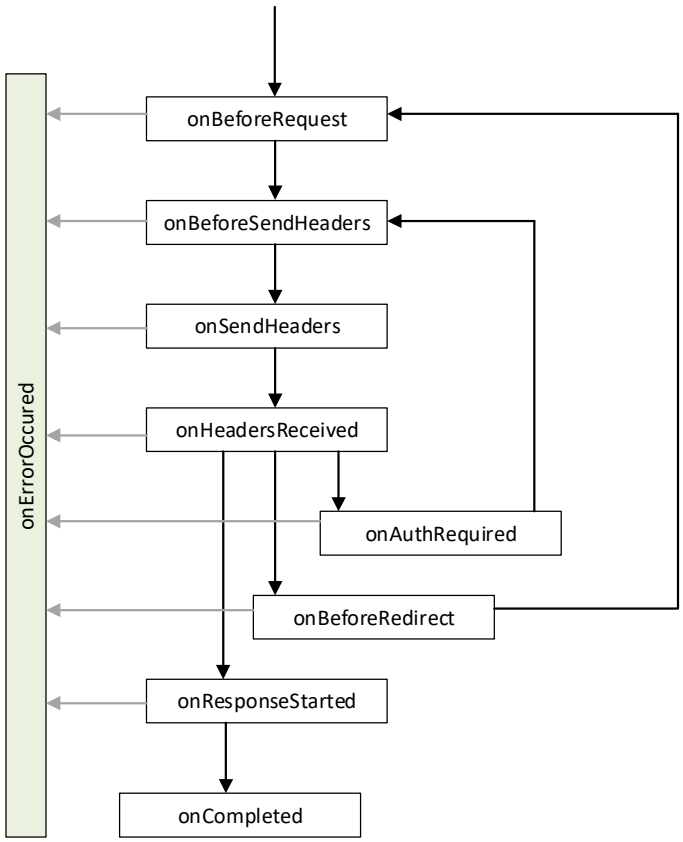


Figure 6-11 the web request life cycle

In terms of operation, the web browser would listen to HTTP request and notifies FP-prevention (browser extension) before sending the HTTP request. FP-prevention would replace the original HTTP request by spoofing some attributes on HTTP header fields (e.g., User-agent) by bringing data from randomness in JavaScript layer to spoof attributes on HTTP header fields. Attempting to maintain correct links of information between HTTP header fields and JavaScript layer will reduce the problem of information paradox.

6.5 Conclude of this chapter

The major objective of this chapter is to clarify the idea behind the proposed countermeasure. Initially, the chapter selects countermeasures that consider the user browsing experience to briefly describe their concepts and displays the weaknesses that they fail to notice. In order to address problems that are overlooked, the FP-prevention browser extension had been proposed in this chapter. Then, the general features of FP-

prevention and the intuition behind the idea of FP-prevention had been described. FP-prevention uses the concept of changing attributes in a systematic approach without relying on the fingerprinting detection part. The idea of FP-prevention seems to be a straightforward countermeasure, but it expects to tackle almost all problems occurring from the current fingerprint countermeasures. This idea originates from the insightful knowledge about how the web browser fingerprinting technique works and how the fingerprint countermeasure works and which weaknesses are ignored by the current fingerprint countermeasures.

(This page intentionally left blank)

Chapter 7 Analysis of FP-prevention

This chapter is divided into three main sections. The first part measures the overhead of FP-prevention compared with an unmodified browser in Section 7.1. In the next section, FP-prevention will be measured on the effectiveness of fingerprinting prevention compared with three countermeasures in order to find which one is the most effective countermeasure. The final part is to measure the level of information paradox in the use of FP-prevention compared with three countermeasures in order to discover which countermeasures are likely to create the side effects.

7.1 JavaScript performance benchmark

Users fully expect that their web browser would be run smoothly and rapidly. Their installed browser extensions in particular should not undermine the functionality of the web browser. Google Chrome extension is mainly built on web technologies such as JavaScript, HTML and CSS (Starov and Nikiforakis, 2017). These technologies should not bring about any side-effects after installing the browser extension.

Since JavaScript is widely used to improve the user experience, the performance of JavaScript is a major concern for modern web browsers. The strong competition among web browsers, recognized as the browser wars (Windrum, 2004), has encouraged a way to measure the JavaScript performance (Gal et al., 2009). Because the browser market share is highly significant to vendors, a valid comparison of JavaScript performance is priceless to both browser vendors and consumers (Ratanaworabhan et al., 2010). Thus, the emergence of many JavaScript performance benchmarks at that time such as Sunspider or V8[73], are intended for measuring the JavaScript performance in order to promote their browsers.

[73] https://developers.google.com/v8/

7.1.1 Experimental setup

The browser performance can significantly affect a user's browsing experience (Ratanaworabhan et al., 2010). In order to measure the execution time, three JavaScript performance benchmarks, Kraken[74], JSBench[75], and SunSpider[76], will be used to measure the average execution time. FP-prevention will be executed 30 times, details of calculating a sample size shown in Appendix D.4. Each time the browser cache will be cleared before running. The PC for this experiment will support Windows 7 Enterprise, 16 GByte of RAM, an Intel Core i7-4770 CPU, with a clock speed of 3.40 GHz. The experimental result will provide the average execution time and the standard deviation. In order to confirm that the execution time is acceptable, an independent t-test is used to calculate the difference between a group with FP-prevention installed and a group without. The results of t-test will suggest significant difference between the two groups in terms of the statistic.

7.1.2 Statistical analysis

The statistical method will be used to assess the significant difference between two groups. The mean value alone is not sufficiently representative of the population. The distribution of the sample should be considered along with other values. The t-test was proposed by Gossett's work (Student, 1908) with the purpose of determining if two groups of a sample are significantly different from each other or not.

T-test can be classified into one sample t-test, two independent t-test and two paired sample t-test. T-test is satisfactory for samples less than 30 (Feng et al., 2017). This research selected the independent t-test because this method is intended for comparing means between two unrelated groups.

The results of two groups, with and without FP-prevention installed, were compared by SPSS version 24. The significance level (α) is set as 0.05 which is employed as cut-off for significance. It means that if p value is less than 0.05, there is a significant difference between two groups. p value < 0.05 was regarded as statistically significant.

[74] https://wiki.mozilla.org/Kraken
[75] http://plg.uwaterloo.ca/~dynjs/jsbench/
[76] https://webkit.org/perf/sunspider-1.0.2/sunspider-1.0.2/driver.html

7.1.3 Result of JavaScript performance

Table 7-1 shows means of execution time (in ms) of three JavaScript performance benchmarks (Kraken, JSBench and SunSpider), along with results of the independent t-test. With the execution time related to the user browsing experience, the difference of execution time between groups needs to be regarded.

Considering only mean value of each JavaScript performance benchmark, it seems insufficient to explain the nature of data. For this reason, the difference of means among groups needs to be tested by statistics. Thus, an independent t-test was conducted to compare the execution time between browsers with and without FP-prevention. The results showed that installing FP-prevention shows an insignificant effect on the performance of Chrome browser, judging from two of three benchmarks.

Considering results of JSBench, there was a significant difference in scores without FP-prevention (M=75.502, SD=1.63) and with FP-prevention installed (M=73.651, SD=1.58); $t(58)=4.448$, $p<0.001$. Even though the result showed a statistically significant difference with $p<0.05$, the mean of the browser with FP-prevention was less than the mean without FP-prevention. Thus, the result suggested that installing FP-prevention shows an insignificant effect on the execution time of the Chrome browser.

Regarding SunSpider, its result is dissimilar to the previous result. Without FP-prevention (M=237.53,SD=4.90) and with FP-prevention (M=237.40,SD=7.87); $t(58)=0.077$,$p=0.939$, it means that the execution time between the two groups is not significantly different ($p>0.05$). Therefore, it indicated that installing FP-prevention shows an statistically non-significant effect on the execution time of the Chrome browser.

The result of Kraken is different from the other benchmarks. It showed significant difference in scores for the browser without FP-prevention (M=1159.716, SD=4.064) and with FP-prevention (M=1161.660,SD=3.326); $t(58)=-2.027$, $p=0.047$. With $p<0.05$, therefore, this result suggests that installing FP-prevention has a statistically significant side effects on the execution time of Chrome web browser.

Table 7-1 comparison of JavaScript performance benchmark

benchmark	Browser	Mean	Standard deviation	t	df	Sig.(2-tailed)
JSBench	Chromium	75.502	1.63	4.448	58	***p < 0.001***
	FP-prevention	73.651	1.58			
SunSpider	Chromium	237.53	4.90	0.077	58	0.939
	FP-prevention	237.40	7.87			
Kraken	Chromium	1159.716	4.064	-2.027	58	***0.047***
	FP-prevention	1161.660	3.326			

7.1.4 **Discussion and conclusion**

A browser speed test is associated with the user experience and can reflect the performance of a web browser. Usually, performance of a web browser can be measured through completion of a predefined list of tasks, mainly using JavaScript benchmark. With the intention to knowing side effects of a web browser with FP-prevention installed, this research attempts to compare a browser speed test between a web browser with and without FP-prevention by using the results of three JavaScript benchmarks' execution time to indicate the performance of a web browser.

The author found that installing FP-prevention did not create any side effects on a web browser, compared with a web browser without FP-prevention. The results of the statistical test suggest that average execution time is similar, with no significant difference. Although one JavaScript benchmark (Kraken) shows that installing FP-prevention has a significant effect on a web browser, the other two JavaScript benchmarks, JSBench and Sunspider, show an effect in the opposite direction, but insignificant.

Therefore, the research determines results of performance of a web browser from the majority of the JavaScript benchmarks. The results of two of three benchmarks suggested that installing FP-prevention has insignificant side effects on the performance of Chrome browser. It means that execution time run by FP-prevention might not induce

significant side effects on the performance of a web browser (additional results provided in Appendix D).

7.1.4.1 Comparing the traditional approach

Usually, many published research, such as FPRandom (Laperdrix et al., 2017) or Privaricator (Nikiforakis et al., 2014b), show only the results of a comparison of mean provided by JavaScript benchmark between a unmodified web browser and modified web browser and then bring results of a comparison to anticipate potential side effects on a web browser. In this research. the author considers that regarding only the execution time of a web browser is not sufficient to establish the fact. Thus, using the repeatability to observe the fact about the performance of a web browser along with using knowledge of statistical test is much preferable in term of academic study, comparing the existing approach. Thus, the difference of this research will observe the average of execution time of two groups provided by each JavaScript benchmark many times in order to use statistical test (student t-test) to suggest the potential of side effects on a web browser, as opposed to the traditional approach.

7.1.4.2 The time constraint

Many published research used different JavaScript benchmarks to measure the execution time. For example, FPRandom (Laperdrix et al., 2017) approves that using a Jetstream benchmark is suitable to test a performance of a web browser. With the time constraint of this research, this research chooses only three JavaScript benchmarks to testing. In order to increase reliability, the idea to increase the number of JavaScript benchmarks to test is necessary for the future work.

7.2 Evaluation of fingerprinting prevention

Nikiforakis et al. (2013) had conducted a survey to quantify the use of browser fingerprinting in the Alexa top 10,000 popular websites[77]. They established that there are two methods that fingerprinting could be commonly used in websites. Firstly, the

[77] https://www.alexa.com/topsites

fingerprinting code is used by advertising syndicators and the resulting fingerprinting ID is sent back to the fingerprinters. This method took place without the first party websites realizing whether fingerprinting code is running on their websites. Secondly, the first party website directly requests the fingerprinting process in order to fingerprint a user.

Normally, each fingerprint company will provide their fingerprint services to many websites and takes the fingerprinting ID from each site, an N-to-1 relationship. Consider, however, the fingerprinting experiments of Mayer (2009), Eckersley (2010), FaizKhademi et al. (2015), Nikiforakis et al. (2014b) and Laperdrix et al. (2015), they used a 1-to-1 relationship between the page running fingerprinting code and the backend placing the results of the fingerprint. In terms of experimental setup, the second method is relatively convenient to measure the effectiveness of fingerprinting prevention compared with the first method.

In order to evaluate how FP-prevention influences the fingerprinters' capability, the 1-to-1 relationship had been selected to simulate the scenario of being fingerprinted in this research. The three fingerprinting services, FingeprintJS[78], PetPortal[79] and Hidester[80], were selected to measure through black box testing by observing the change of fingerprinting ID. The black box testing, well-known as behavioural testing, will be used in this research because this method is not necessary to know all details of created software (Limaye, 2009). This decision was made because the secrecy of fingerprinters' code makes it difficult to find how many attributes are used for fingerprinting.

Therefore, for evaluation of fingerprinting prevention in this experiment, the web browser will be forced to visit the fingerprinting page directly because the change of fingerprint ID while visiting each time, normally related to achievements of the fingerprint tracking, will be observed straightaway.

7.2.1 **Experimental setup**

In order to conduct the study on the effectiveness of fingerprinting prevention, the research had combined manual and automated analysis for this experiment with the

[78] https://github.com/Valve/fingerprintjs2
[79] https://fingerprint.pet-portal.eu/
[80] https://hidester.com/browser-fingerprint/

purpose of observing the overall trend of fingerprinting ID after the web browser had visited the fingerprinting page multiple times. This experiment used Firefox version 47.0.2 and Chrome version 65.0.3325.181 for installing fingerprint countermeasures.

In order to automate the process of visiting a fingerprinting site, the iMacros extension[81] had been selected to perform the repetitive task, for visiting each fingerprinting site multiple times. iMacros is intended for performing repetitive tasks, such as visiting a number of websites, clicking on links, checking on the same websites every day, submitting to web search engine or examining websites, and are made available on the Internet both Chrome and Firefox. For this research, iMacros will be used to collect fingerprints given and then analyze them. This research has developed a script running on iMacros as shown in Figure 7-1. The web browser with installed fingerprint countermeasure will be forced through iMacros to visit the fingerprinting page and wait 10 seconds for loading scripts and wait over 15 seconds in case a website is unloaded. When the fingerprinting site is completely loaded, the fingerprinting ID given will be recorded into the comma-separated values[82] (CSV) file automatically.

Script used for automating data collection

```
For a number of a visiting website
Open the website in the tab
refresh
waiting 10 seconds for  loading a website
waiting over 15 seconds if a websites is not loaded
```

Figure 7-1 pseudocode used for automatic task on iMacros

In respect of manual analysis, the research will collect fingerprints provided manually with the main reason that FP-Block will change attributes when a user must change URL of

[81] https://chrome.google.com/webstore/detail/imacros-for-chrome/cplklnmnlbnpmjogncfgfijoopmnlemp?hl=en
[82] http://www.ietf.org/rfc/rfc4180.txt#page-1

the website only. With this reason, FP-block will regenerate a web identity manually after visiting the fingerprinting site every time. This method is absolutely essential because attempting to modify FP-Block's code in order to run an automated process without understanding how FP-Block's code works leads to obtaining false results.

For selecting fingerprinting providers, the research had selected *FingerprintJS2*, *Hidester* and *PetPortal* to conduct this experiment. The research measured how FP-prevention stands against these fingerprinting providers by comparing with Stop fingerprinting[83], Fireglove[84], and FP-Block[85]. The brief details of fingerprinting providers is described below.

FingerprintJS2:[86] it is a popular open-source fingerprinting library which uses 27 attributes to fingerprint a user. The research selects FingerprintJS2 because it uses fonts, JavaScript object (navigator and screen object), plugins and canvas for fingerprinting.

Hidester: it is a company to provide VPN services. This company established VPN servers across the world, such as Brazil, USA, France, Germany, Japan and Taiwan, in order to assist a user to stay anonymous on the Internet. Hidester has provided a fingerprinting page through which users can test their fingerprint[87].

PetPortal:[88] this is a fingerprinting website which is developed by Boda et al. (2012a). This fingerprint website required to improve the stability of fingerprinting by using more attributes (e.g., IP address) in order to be able to track a user across-browser and find an alternative way of accessing user's fonts rather than relying on APIs provided by a plugin.

Before presenting the results of these experiments, more details will be explained about some fingerprinters not used in this experiment.

Bluecava[89]: it is the well-liked fingerprinting company that employs different attributes and methods to fingerprint a user. Usually, Bluecava provides the opt-out page

[83] https://addons.mozilla.org/en-GB/firefox/addon/stop-fingerprinting/
[84] https://fingerprint.pet-portal.eu/?menu=6
[85] https://addons.mozilla.org/en-GB/firefox/addon/fp-block/
[86] https://github.com/Valve/fingerprintjs2
[87] https://hidester.com/browser-fingerprint/
[88] https://fingerprint.pet-portal.eu/
[89] https://bluecava.com/

in which users can select to stop sharing their fingerprint. However, Bluecava's opt-out page currently is unworkable without any explanation from Bluecava.

Panopticlick[90]: even though the Pantopticlick project galvanised users around the world to pay attention to the problems of hidden tracking through the browser fingerprinting technique, it yields results of testing fingerprint in the form of a statement such as "*Your browser fingerprinting appears to be unique among 1,471,775 users tested so far*". The result in the form of this statement cannot assist research to compare the fingerprinting ID given from multiple visits.

7.2.2 Establishing set of criteria.

There are few ways to establish that a user is likely being tracked by the browser fingerprinting technique. This research had established these criteria in order to judge the fingerprinting ID provided by this experiment.

- **Showing the same consecutive fingerprinting ID:** if the fingerprinting ID given shows the same consecutive fingerprinting ID, it clearly shows that a user is being tracked by the browser fingerprinting technique.
- **Showing null value:** the main purpose of fingerprint countermeasures based on randomization or spoofing technique is to deceive the fingerprinters with false information. Obtaining the null value opposes the main objective of fingerprint countermeasures. Even though obtaining null value can be interpreted that fingerprint countermeasure can deter the process of creating fingerprinting ID, this research will determine that null value is an undesirable value which should be interpreted as inability to block the fingerprint tracking.
- **Showing the duplicate fingerprinting ID:** unlike obtaining the same consecutive fingerprinting ID, this case means that the fingerprinting ID given shows the same fingerprinting ID during visits of the fingerprinting page multiple times. Obtaining the same fingerprinting ID many times can imply that users are likely to be tracked through the browser fingerprinting technique because of leaving a trace for tracking

[90] https://panopticlick.eff.org/

to fingerprinters. In contrast, showing a unique fingerprinting ID during each visit to the fingerprinting page shows ability to block the fingerprint tracking according to the main objective of the countermeasure.

7.2.3 **Results of fingerprinting prevention**

The results of this set of experiments are shown in Figure 7-2, 7-3 and 7-4 below. The research had divided the range of observing the fingerprinting ID into five series, ranging from 0 to 500 visits to the fingerprinting page in increments of 100. The browser with installed countermeasure will be forced to visit each fingerprinting page in line with the defined range. The expected result is to observe the number of fingerprinting IDs provided by a fingerprinting page.

The bar graph below shows the number of duplicate fingerprinting IDs and unique fingerprints during visiting the fingerprinting page, ranging from 100 to 500 times. In respect of the bar graph, the x-axis represents the number of times visiting the fingerprinting page and whether the fingerprinting ID is unique or duplicated, while the y-axis represents the number of duplicate and unique fingerprinting IDs provided by the fingerprinting providers.

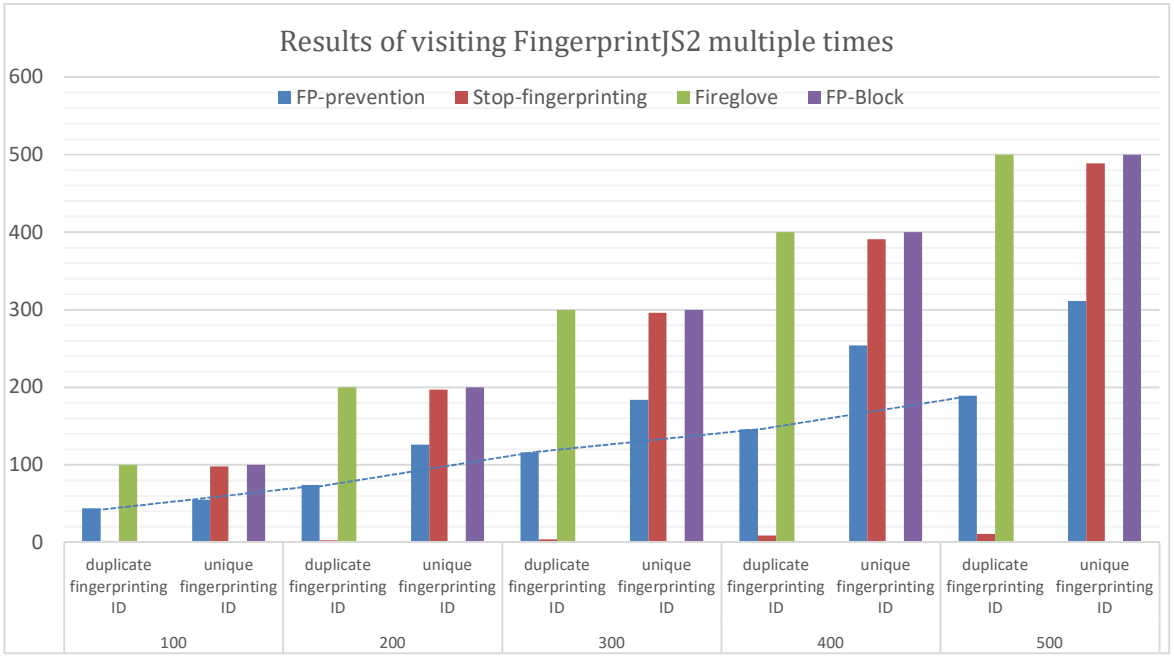


Figure 7-2 results of visiting FingerprintJS2

FingerprintJS2: for the results of FingerprintJS2 in Figure 7-2, overall the highest number of duplicate fingerprinting IDs is Fireglove, it showed only the null value to FingerprintJS2. In terms of null value, it means that FingerprintJS2's code cannot generate any fingerprinting ID. However, this research determines that null value is ineffective in obfuscating fingerprinters and makes the web browser stand out among the crowd. The fingerprinters may find additional measures to fingerprint user further.

The second highest number of duplicate fingerprinting IDs is FP-prevention. It started to show 44 duplicate fingerprinting IDs with 100 times of visiting FingerprintingJS2 and showed 181 duplicate fingerprinting IDs and 311 unique fingerprinting ID with 500 times of visiting FingerprintingJS2. With regard to the number of duplicate fingerprinting IDs produced by Stop-fingerprinting, it shows only 11 duplicate fingerprinting IDs with 500 times of visiting FingerprintJS2, while FP-Block showed the most effective countermeasures against tracking of FingeprintJS2 by yielding 500 unique fingerprinting IDs with 500 times of visiting FingerprintJS2.

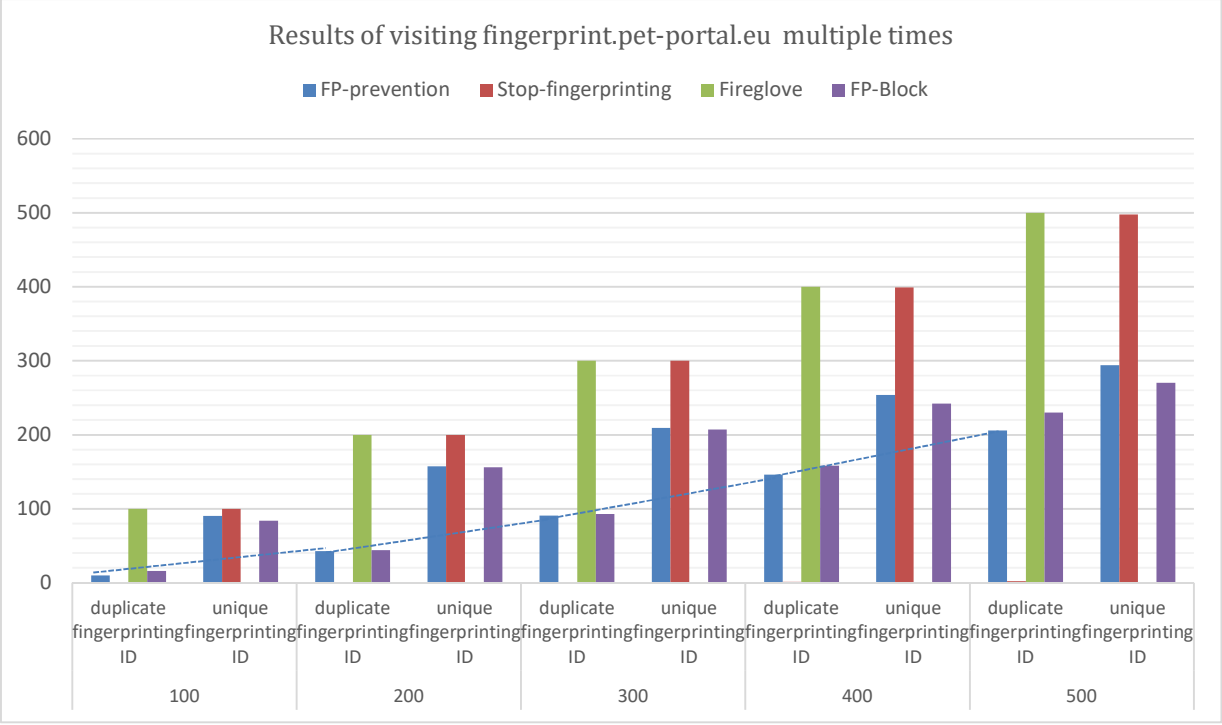


Figure 7-3 results of visiting fingerprint.pet-portal.eu

PetPortal: for the results of PetPortal in Figure 7-3, overall they are dissimilar to the results of visiting FingerprintJS2. Even though Fireglove still stands the most ineffective countermeasure at deceiving the fingerprinting providers, the second most ineffective countermeasure appears to be FP-Block. Results of the number of its duplicate fingerprinting IDs differs from visiting FingerprintJS2 in a number of respects. FP-Block showed 16 duplicate fingerprinting IDs with 100 times of visiting PetPortal and 230 duplicate fingerprinting ID with 500 times of visiting PetPortal, whereas it showed no duplicates when visiting FingerprintJS2. With regard to FP-prevention, at beginning FP-prevention showed a gradual increase in the number of duplicate fingerprinting IDs at 10 fingerprinting IDs with 100 times of visiting PetPortal before dramatically increasing to 43, 91, 146 and 206 for 200, 300, 400, 500 visits, respectively. Even though FP-prevention shows poor performance, it still ranks as the second most effective countermeasure against PetPortal. Stop-fingerprinting is the most effective countermeasure by showing 498 unique fingerprinting IDs and 2 duplicate fingerprinting IDs with 500 times of visiting PetPortal.

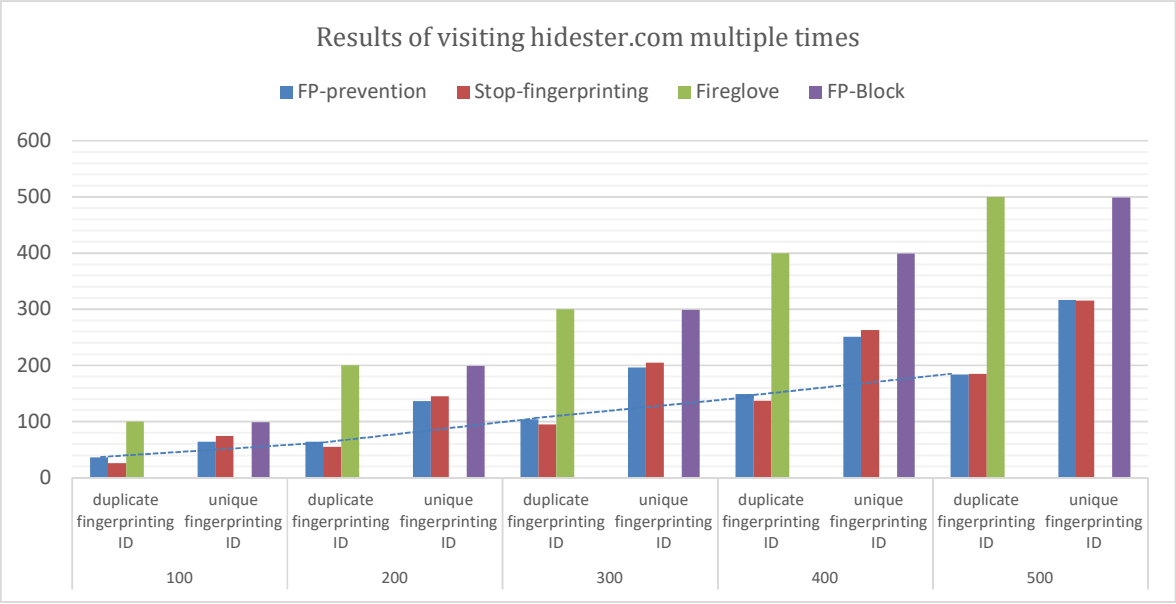


Figure 7-4 results of visiting hidester.com

Hidester: Lastly, Figure 7-4, illustrates the results of the experiment against Hidester. It is evident that FP-Block showed the most effective countermeasure against Hidester by

providing only one duplicate fingerprinting with 500 times of visiting Hidester. The number of duplicate fingerprinting IDs between FP-prevention and Stop-fingerprinting showed slight differences of their performance. FP-prevention seems to show a poor performance in the beginning by providing the duplicate fingerprinting greater than Stop-fingerprinting. However, when the number of visits increased, Stop-fingerprinting appeared to show worse performance than FP-prevention by providing 185 duplicate fingerprinting IDs with 500 times of visiting Hidester, while FP-prevention showed 184 duplicate fingerprinting IDs, less by only one. Fireglove still showed the worst performance similar to all previous results.

Summary: a practical experiment of this research showed the difference of countermeasures against three fingerprinting providers. FP-Block showed the most effective countermeasures against two of three fingerprinting providers. The second most effective countermeasure is Stop-fingerprinting, while FP-prevention is ranked as the third most effective countermeasure. Fireglove showed the worst performance against three fingerprinting providers compared with four fingerprint countermeasures.

7.3 Assessing the information paradox

In the previous section, the research demonstrated the ability of FP-prevention to combat fingerprinting providers by comparing it with other countermeasures. In this section, the paper assesses the level of information paradox of four countermeasures scrutinized in the previous section.

The four countermeasures attempt to address the problems of fingerprinting prevention by providing false information to fingerprinters. However, changing attributes may lead to inconsistency of attributes and make the web browser stand out.

7.3.1 Methodology of conducting information paradox

In order to assess the information paradox, the author needs to pretend to be a genuine fingerprinter. The author has established a new fingerprinting website for this

assessment,[91] as the old website[92] used in the empirical study encountered problems with database backup and export of data. Over 1,000 participants were engaged in these experiments through visiting a new fingerprinting website where their fingerprint will be collected and stored in a database system. The participants' information assists the research to establish the metrics to be able to distinguish between a normal web browser and an abnormal web browser, with the aim of assessing the information paradox.

7.3.2 **Ethics**

The Ethics and Research Governance Online of the University of Southampton has approved this study (ethics number ERGO/FPSE/41395). The participants interested in testing their fingerprint through a web browser fingerprinting technique are required to read basic information about the web browser fingerprinting technique on a provided webpage. Then, they are required to tick a checkbox confirming that they consent to taking part in this survey. Finally, they need to click on a button to agree to test their web browser. Further details and a Figure of the main web page can be found in Appendix E.

7.3.3 **Data collection**

Participants for this part of the study have been invited through many channels. This includes a group of lecturers and students at the department of computer technology at Rajamangala University of Technology Isan Surin campus (CP-Rmuti) in Thailand, as well as friends and colleagues at the University of Southampton.

When participants click on a button to agree to test their fingerprint, all fingerprinting processes will be run automatically on the client side, and will then send users' information and fingerprinting ID back to be stored in the server's MySQL database through Asynchronous JavaScript and XML (AJAX). The screen will display the results of the fingerprinting whether or not a user is unique among all fingerprinted users (Figure of fingerprinted user shown in Appendix E), including showing the level of information paradox of a user's computer. Many participants are participating in the survey online, including participants using smartphones (e.g., iPhone or Samsung) or desktop computers

[91] http://www.pleasefingerprintme2.org/index.php
[92] http://www.pleasefingerprintme.org/

7.3.4 **Establishing Criteria for measuring information paradox**

The meaning of information paradox is that obtained information shows incomplete information or shows impossible combinations. This makes the obtained information stand out from normal information.

Table 7-2 Metrics for calculating levels of information paradox

	Metric	**Description**
Incomplete coverage	Metric 1	A number of attributes show undefined value more than two attributes
	Metric 2	A number of attributes show empty value more than two attributes
Mismatch between header	Metric 3	A value of User_agent shows inconsistencies with the User-agent request in HTTP Headers.
	Metric 4	A value of navigator.languages shows inconsistencies with Accept_Language in HTTP headers.
Inconsistency between related attributes	Metric 5	A value of navigator.language in navigator object shows inconsistency with navigator.languages. For instance, Language shows en-us but Languages shows fr-ca.
	Metric 6	The User-agent shows inconsistency with operating system
	Metric 7	Mismatch between the operating system and the screen display. For instance, iPhone shows width and height of screen at 1920x1080 pixel.

According to the results of Nikiforakis et al. (2013)'s experiment, browser extensions used for preventing invasions of a user's privacy by hiding their real identity had three problems with information paradox, namely: incomplete coverage of navigator object; inconsistencies between related attributes; and the mismatch between information on HTTP header and information on JavaScript layer . Therefore, this research observed the

characteristics of collected data in the database through participants visiting a fingerprinting website, and then splits the three problems with information paradox established by Nikiforakis et al. (2013) into seven metrics, which are indicators for measuring the levels of information paradox of users visiting a fingerprinting website

As can be seen in Table 7-2 (above), there are seven metrics for measuring whether user information is normal or abnormal. These metrics derive from observing the user data collected through the online survey. A set of metrics were then designed that correspond to the nature of the users' data.

Metric 1: With regard to the user data collected in the database, they did not display an undefined value for more than two attributes. Therefore, if any user information shows undefined value for more than two attributes, it implies that the obtained user information seems to be fake information.

Metric 2: Regarding the user data collected in the database, they did not display an empty value for more than two attributes, similar to the undefined value in Metric 1. Thus, if any user information shows an empty value for more than two attributes, it implies that the obtained user information seems to be unreliable.

Metric 3: Normally, navigator.userAgent is always equal to the User-agent request header in HTTP headers. Thus, mismatch between navigator.userAgent and User-agent request header in HTTP headers is a sign of abnormal information.

Metric 4: Typically, navigator.languages is consistent with Accept-Language on the HTTP headers. Thus, mismatch between navigator.languages and User-agent Accept-Language on the HTTP headers is a sign of abnormal information.

Metric 5: Usually, the navigator.languages returns an array of user's preferred languages. The returned array will be sorted by favourite with the most preferred language first as shown in Figure 7-5 (below). Thus, if the first array of navigator.languages shows a different language to navigator.language, it implies that the obtained language value seems to be fake information.

```
navigator. language  // “en-US”

navigator.languages  // [‘’en-US”, “th”,”fa”]
```

Figure 7-5 the right relationship between navigator.language and navigator.languages

Metric 6: Typically, the User-agent information and operating system information will be associated with one another. For example, Internet Explorer is associated with Windows. It is impossible to see Internet Explorer running on a Macintosh operating system. This research has established a way to check the relationship between User-agent and operating system as follows:

Firefox: Windows, Macintosh operating system and Linux.

Chrome: Windows, Macintosh operating system and Linux.

Opera: Windows, Macintosh operating system and Linux.

Internet Explorer: Windows.

Safari: Windows and Macintosh operating system.

Metric 7: The obtained information should show the possible link between screen information (width and height) and operating system. For example, iPhone shows use of a screen at 1920 x 1080 pixels, making it impossible to accept that the information about the obtained operating system and screen is genuine. The research established that a smartphone’s screen (Samsung or iPhone) should not be over 900 x 900 pixel.

- **Attributes used for checking information paradox**
  - **HTTP header field** (2 attributes)**:** *User-agent* and *HTTP accept-language*
  - **Navigator object** (17 attribute): *appCodeName, appName, appVersion, cookiedEnable, doNotTrack, language, languages, maxTouchPoints, mimeTypes, online, platform, plugins, product, productSub, Useragent, vendor* and *vendorSub.*

- **Screen object** (9 attributes)**:** *ColorDepth, PixelDepth, width, height*, *bufferDepth, availLeft, availTop, availHeight and availWidth.*
- **List of fonts** (1 attribute):
- **List of plugins** (1 attribute):

In terms of measuring the level of information paradox, this value derives from the sum of all metrics into the levels of information paradox. Typically, a user who visits the fingerprint page will be fingerprinted and their user information analysed. The analysis of user data will be calculated according to the metrics in Table 7-2 and will be displayed in a form of levels of information paradox to the user being tested, as shown in Figure 7-6.

7.3.5 **Experimental setup**

This experiments consisted of manual and automated analysis. In part of the automated analysis, Fireglove, Stop fingerprinting and FP-prevention will be forced to visit new fingerprint websites 100 times through iMacros[93]. All extracted metrics will be stored on the database server automatically. Regarding web identity, FP-block will not change attributes if a user does not change the URL of the website. In addition, FP-block is designed to block almost all attributes in the first instance before making the decision to change attributes by a user. This makes it a problem to store attributes in the database server. Furthermore, since the loading of script by FP-block is sometimes uneven, other repetitive programs (such as Tinytask[94]) cannot be replaced. Therefore, the FP-block will regenerate a user's identity manually.

The four fingerprint countermeasures would be validated by visiting a new fingerprinting website 100 times, with the purpose of observing the overall trend of changed attributes. The information of a user computer will be extracted on every visit through metrics. The user's screen will show only the levels of information paradox, as shown in Figure 7-6, while all results of the extracted metrics will be stored on the database

[93] https://imacros.net/
[94] https://www.tinytask.net/

automatically. The research determines that the levels of information paradox range from zero to seven levels, as shown in Figure 7-6.

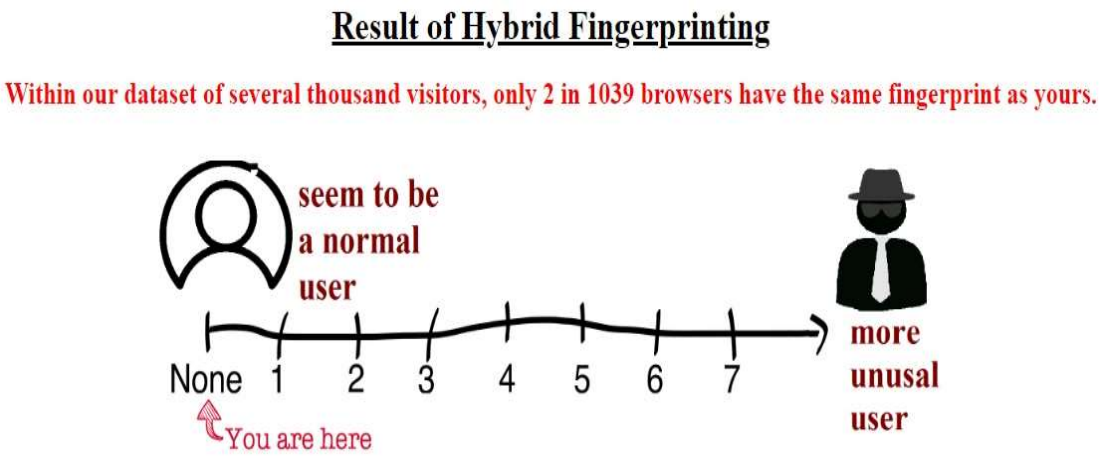


Figure 7-6 the result of levels of information paradox in case of a normal user

7.3.6 **Results**

The results of this experiment have shown the problems with the changing attributes of each countermeasure. Looking at Figure 7-7, the y-axis represents the number of detected attributes from visiting a fingerprinting page by a web browser, while the x-axis represents tested countermeasures, namely Fireglove, FP-block, Stop-fingerprinting and FP-prevention. Each metric will be illustrated through one graph and the final graph shows levels of information paradox for all tested countermeasures.

7.3.6.1 Result metric 1 (a number of attributes showing undefined value)

In Figure 7-7, the bar chart illustrates a number of undefined values deriving from the database. Overall, FP-block yields the highest number of undefined values at 728, almost double that of Fireglove, which yields 400. Stop fingerprinting provides 200 undefined values, followed by FP-prevention at 100. It can be seen that Stop fingerprinting and FP-prevention provides the normal web browser information to the browser fingerprinting technique (not over a number of defined metric 1), while FP-block provides a large number of undefined value, making it difficult to accept that they are genuine attributes.

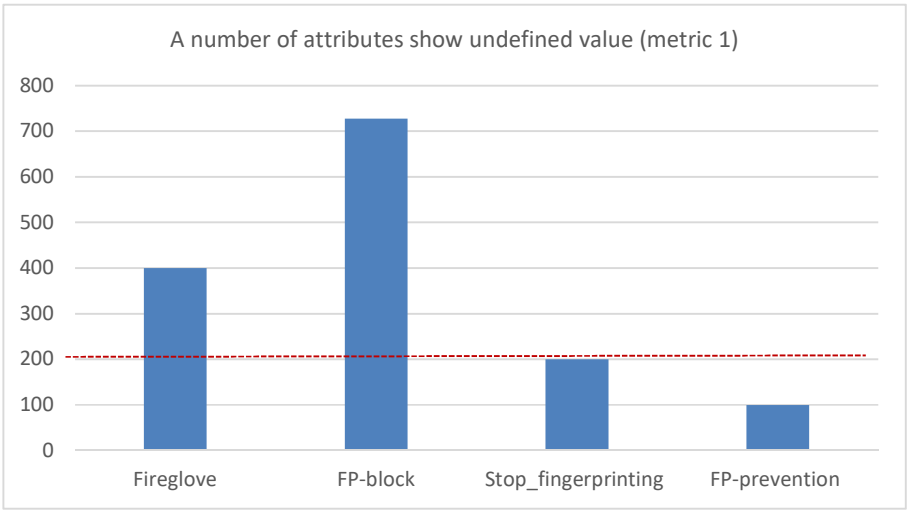


Figure 7-7 a number of attributes show undefined value

7.3.6.2 Result metric 2 (a number of attributes showing empty value)

In Figure 7-8, overall, it can be seen that the result is dissimilar to the previous result. It appears that Fireglove yields the highest number of empty values at 300, followed by Stop fingerprinting at 200. FP-block provides a small number of empty values at 119, while FP-prevention shows a 40 empty values. To sum up, only Fireglove shows abnormal information to a web server while other countermeasures did not break the threshold. This means that they provide user information to a web server within a normal range.

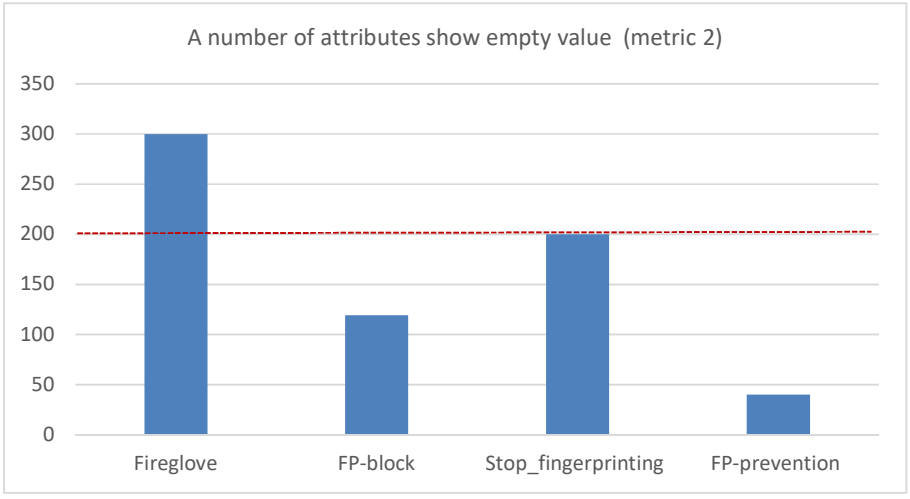


Figure 7-8 a number of attributes show empty value

7.3.6.3 Result metric 3 (inconsistency between User-agent request in HTTP Headers and navigator.userAgent)

In Figure 7-9, three countermeasures do not show inconsistency between User-agent request in HTTP Headers and navigator.userAgent, with the exception of FP-block. It shows inconsistency at two from 100 visits to a fingerprinting website.

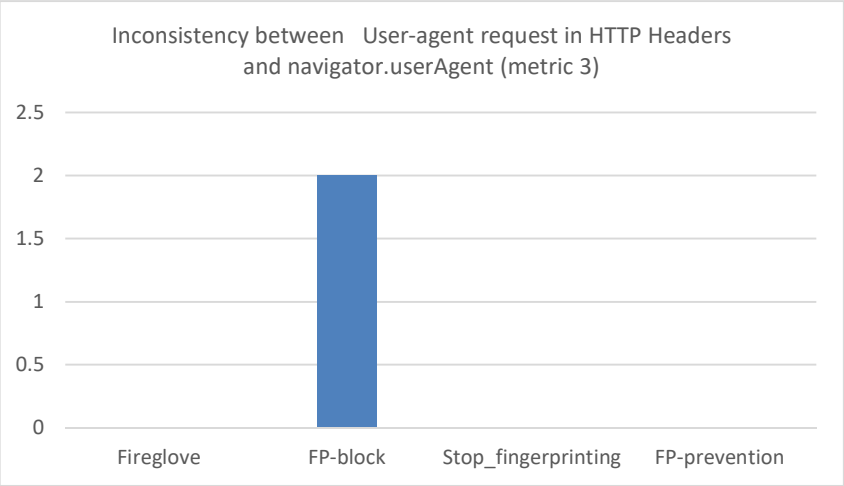


Figure 7-9 inconsistency between User-agent request in HTTP Headers and navigator.userAgent

7.3.6.4 Result metric 4 (inconsistency between Accept_Language in HTTP headers and navigator.languages)

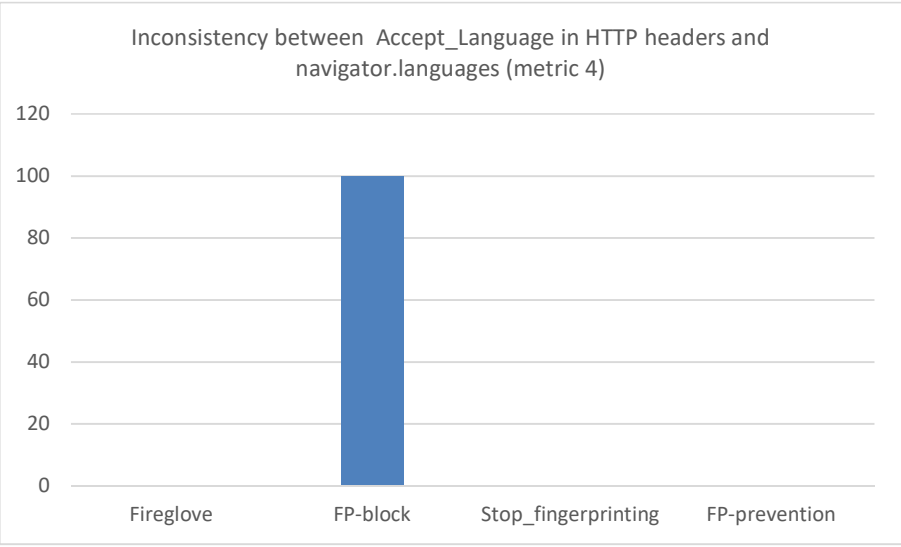


Figure 7-10 inconsistency between Accept_Language in HTTP headers and navigator.languages

In Figure 7-10, only FP-block shows inconsistency between Accept_Language in HTTP headers and navigator.lang3uages by showing 100 from visiting the fingerprinting website 100 times. It clearly shows that changing the attributes of FP-block did not consider information about Language on HTTP header at all. Since Fireglove and Stop fingerprinting did not select this attribute for change, it is not surprising that they did not show inconsistency between Accept_Language in HTTP headers and navigator.languages. Similarly, FP-prevention had considered Language attribute to be problematic. Therefore, FP-prevention has no problem with this metric.

7.3.6.5 Result metric 5 (inconsistency between navigator.language and navigator.languages)

Figure 7-11 clearly shows that Fireglove and FP-block have substantial problems with this metric, while Stop fingerprinting and FP-prevention did not show any problems at all. Fireglove has the most problems with this metric, with inconsistency occurring on every visit to a fingerprinting website, while FP-block performs similarly to Fireglove, showing inconsistency on 96 out of 100 visits.

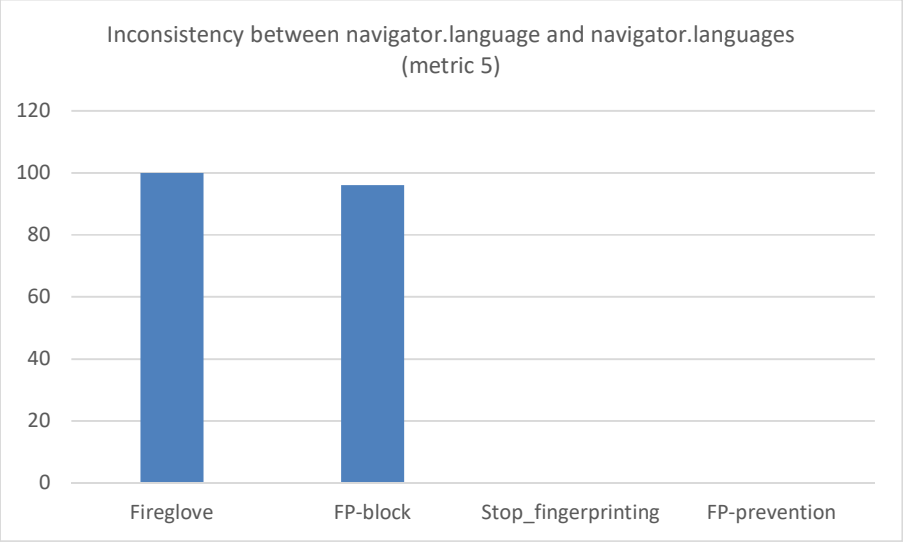


Figure 7-11 inconsistency between navigator.language and navigator.languages

7.3.6.6 Result metric 6 (inconsistency between User-agent and operating system)

In Figure 7-12, overall, this metric makes it clear that Fireglove is extremely careless in changing attributes. It yields the highest number of inconsistencies between User-agent and operating system, with 82. Meanwhile, FP-block presents a small number of inconsistency between User-agent and operating system at 2, meaning that it compares very favourably with Fireglove. On the other hand, Stop_fingerprinting did not change User-agent and operating system at all. Thus, it does not show abnormal information for this metric. While FP-prevention handles this attribute by the FP-prevention model, it is not surprising that FP-prevention did not present any effects with this metric.

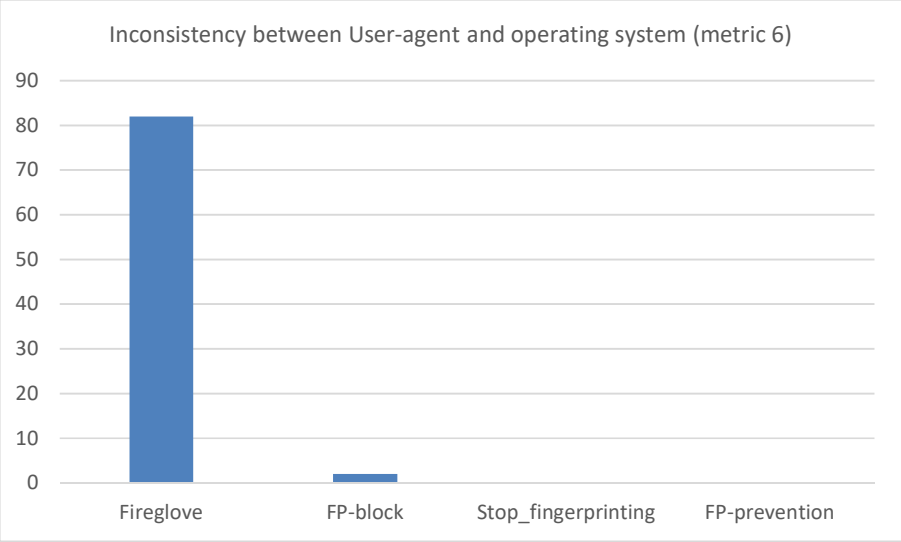


Figure 7-12 inconsistency between User-agent and operating system

7.3.6.7 Result metric 7 (inconsistency between operating system and the user screen)

Regarding Figure 7-13, Fireglove still stands the worst countermeasure for providing abnormal combinations. It shows a modest level of inconsistency between operating system and the user screen at 14, while the remaining countermeasures would be prudent to use for the selection of the right attributes for randomness. Therefore, there are not any problems with this metric.

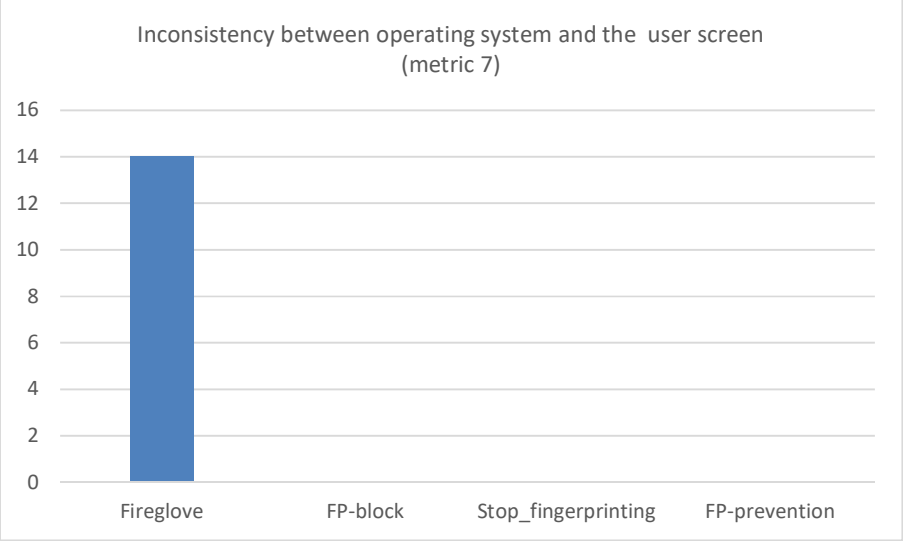


Figure 7-13 inconsistency between operating system and the user screen

7.3.6.8 Result the average levels of information paradox

The average level of information paradox is derived from the sum of all seven metrics after visiting the fingerprinting website 100 times. Usually, the level of information paradox will be shown on the user screen automatically after a user visits a fingerprinting website.

Regarding Figure 7-14, it is evident that Fireglove presents the highest number of levels of information paradox, showing 3.96 levels. The second highest number of levels of information paradox is the FP-block, showing at 3.02 levels, while Stop fingerprinting and FP-prevention did not show any signs of information paradox to the web server at all.

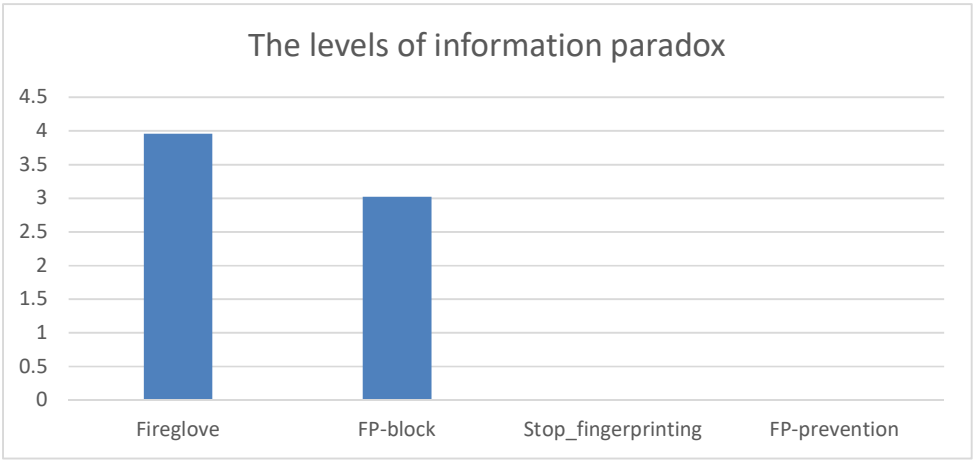


Figure 7-14 the levels of information paradox

To sum up: the inattention to selecting the right attribute for randomization is a main drawback of Fireglove, including changing attributes arbitrarily. These factors are responsible for providing more information paradox compared with other countermeasures. While FP-block is relatively careful about the data used for randomization by attempting to change attributes unrelated to smartphones, it makes a mistake by its inability to hide the related attribute, especially the Language attribute, as well as changing attributes with an excessive number of undefined values. In addition, regarding the concept of web identity of FP-block, it makes their identity stand out from the crowd – it is easier to fingerprint compared with Fireglove even though it has the second highest level of information paradox. Stop fingerprinting, meanwhile, changes only a few attributes such as font attributes. Thus, it is no wonder that this countermeasure still provides the normal combination, while FP-prevention changes attributes by FP-prevention model and chooses the right attribute for randomness. Therefore, it does not show signs of information paradox to a far greater extent than other countermeasures.

7.4 Conclude this chapter

This chapter presents the results of three studies: the performance of a web browser after installing FP-prevention; the effectiveness of fingerprinting prevention; and levels of information paradox. Considering the results of JavaScript performance after installing FP-prevention, the author used three JavaScript performance benchmarks to measure the execution time between installing FP-prevention and uninstalling FP-prevention. The statistical analysis of t-tests was used to determine whether two sets of data are significantly different or not. The statistical result suggests that installing FP-prevention does not have a significant impact on the performance of a web browser compared with uninstalling FP-prevention.

In terms of effectiveness of fingerprinting prevention, four countermeasures were validated by visiting three fingerprinters many times. The overall trend showed that FP-Block is the most effective countermeasure against two of three fingerprinting providers, followed by Stop fingerprinting. FP-prevention proved to be the third most effective countermeasure against fingerprint tracking, while Fireglove ranked as having the worst performance in fingerprinting prevention.

As regards the problem of information paradox, the four countermeasures must visit the fingerprint website many times. The results of visiting the fingerprinting website are analysed for their levels of information paradox through the seven metrics. The results of the analysis confirmed that Fireglove showed the highest levels of information paradox of the countermeasures tested. The second highest levels of information paradox belonged to FP-block, showing slightly different levels from Fireglove. Stop fingerprinting and FP-prevention did not display any problems with information paradox.

Chapter 8 User Experience Analysis

The goal of this chapter is to measure the user experience and to characterize the methodology of the case study conducted in the department of computer technology at Rajamangala University of Technology Isan Surin campus in Thailand. This chapter is arranged as follows: The background of organization where the survey was conducted provided in Section 8.1 and Section 8.2 then provides the general background of user experience and model used to measure the user experience. Section 8.3 describes a methodology for conducting the user experience survey in order to discover which countermeasures show the highest score of the user experience.

8.1 Background of the case study

The department of computer technology at Rajamangala University of Technology Isan Surin campus (CP-Rmuti) had been selected for this case study. This CP-Rmuti was established in 2006 and is one of four campus locations of the Rajamangala University of Technology Isan, located in Surin province. CP-Rmuti is responsible for producing graduates in the field of computer technology with quality and ethics, as well as fulfilling the requirements of the country by increasing the number of hands-on graduates in the field of computer technology. Currently, CP-Rmuti has also a firm commitment to produce excellent quality research in order to serve the country.

CP-Rmuti was selected as there is a sufficient number of students and staff with skills in the use of a computer for conducting the survey. In addition, CP-Rmuti has an adequate number of computers in each laboratory that are already connected with the Internet. These factors would assist the research to conduct the survey. Details about the survey are described in the following sections

8.2 Background of user experience

User experience is an important component in the software system and plays a crucial role in user attraction to use the created software. Currently, many published research papers have applied user experience to many areas such as creating a game online

(Huang et al., 2017), increasing sales in an online product (Nambisan and Watt, 2011) and so on.

In spite of having considerable benefits to the software industry, the definition of user experience is extensively used in many different explanations. ISO 9241-210 standard mentions that it is a "*User experience includes all the users' emotions, beliefs, preferences, perceptions, physical and psychological responses, behaviours and accomplishments that occur before, during and after use*"[95]. Hassenzahl (2008) disagreed that "*all*" is ambiguous and relatively broad as it could be interpreted as "everything". According to Jetter and Gerken (2007), they had the opinion about the definition of user experience that it is impractical to serve every possible feeling and need. Similarly, Law et al. (2009) had reported the result of a survey about the view of 275 researchers and practitioners from academia and industry, that they had different views about the definition of user experience, depending on years of their experiences.

With a broad definition of user experience, it is difficult to select which methods are suitable for measuring it. In order to create a possibility of measuring the user experience, this research had adapted the pleasure-arousal-dominance (PAD) model from Huang et al. (2017), as shown in Figure 8-1. Normally, this PAD model consists of seven factors: functional experience, hedonic experience, social experience, pleasure, arousal, and dominance. However, the PAD model is designed for measuring user experience of a game online. Thus, some metrics would be removed because of irrelevance to measuring user experience of a web browser. The arousal was removed because excitement is not necessary to using a web browser. The metric of functional experience is functionalities of a web browser such as displaying a web page correctly and plugins available to use. The hedonic experience measures feeling and satisfaction of a user. Social experience is measured by asking a user when needs to communicate other users. Pleasure is the feeling after using a countermeasure. Dominance is non-hesitancy in the web browser being used. Finally, Word of Mouth (WOM) is measured by asking users if they would like to share their good experience to other people.

[95] https://www.iso.org/obp/ui/#iso:std:iso:9241:-210:ed-1:v1:en

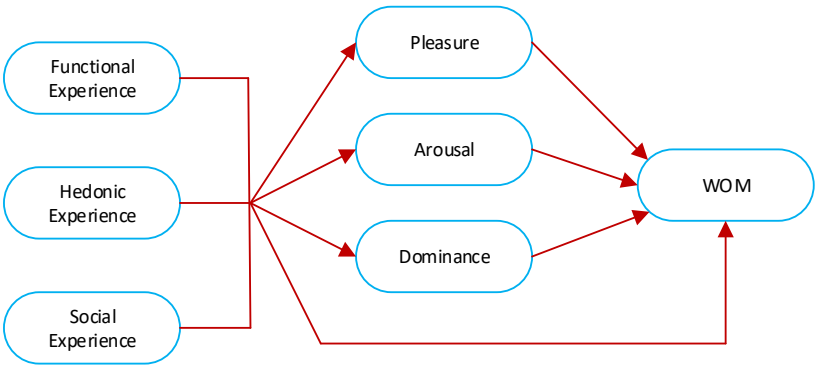


Figure 8-1 pleasure-arousal-dominance (PAD) model

8.3 Methodology for conducting the user satisfaction survey

With respect to the first step of the user satisfaction survey, the survey would be conducted in CP-Rmuti. The result of the user satisfaction survey would assist the research to learn how the user feels to countermeasure. It is indispensable to survey because use of other methods is not certain to measure almost all side effects of user experience.

8.3.1 Questionnaire design

The questionnaire designed consisted of two sections. In the first section, the participant would be asked about general information such as study program (regular or special program), gender, academic year. The second section would ask participants about perceptions and attitudes to the fingerprint countermeasure being used through metrics of PAD model.

The six metrics were chosen from Huang et al. (2017) in order to measure the user experience. These metrics are adapted from PAD model (Huang et al., 2017), details of each metrics shown below.

1. *Functional Experience:* functionalities of a web browser, display of a web browser
2. *Hedonic Experience:* satisfaction of an Internet user and feeling
3. *Social experience:* ability to use a web browser to communicate with other users
4. *Pleasure:* feeling happy after using a web browser

5. Dominance: ability to control a web browser
6. Word of Mouth: willingness to share good experience to other people

Each metric consists of a number of questions representing factors of user experience. There are six questions related to functional experience. There are two questions connected with the social experience and pleasure. Hedonic experience and dominance have two questions. Lastly, there are two questions with regard to word of mouth.

8.3.2 **Consistency of translation between Thai and English version**

It is essential to translate from the English Language into the Thai Language because participants who engaged with this survey are Thai students. For this stage, Thai native speakers who are studying at the University of Southampton would be invited to ask whether the translated questionnaire is consistent with Thai language or not. The research used three Thai native speakers for checking the consistency of language and verifying that the translated questionnaire is easily understandable by Thai people along with maintaining the same meaning of the English version.

For assessment of the consistency of language, three Thai native-speakers were invited to rate a score of consistency between two versions (English and Thai Language). A rating of +1 (consistent), 0 (uncertain about consistency of language) or -1 (no consistency). In case of rating 0 or -1 to a question or statement, they were also asked about additional comments in order to improve the question or statement.

Afterward, the score of each question and statement would be calculated into the total consistency score by adding the rating of each Thai native-speaker. The results of the score would be shown in Appendix C. All Thai statements and questions recommended will be applied to the questionnaire and all consistency scores are higher than two.

8.3.3 **Sample size estimation**

This research uses G*Power application with the aim of calculating an expected sample size (Faul et al., 2009). The calculated sample size for this experiment is 112, details of setting shown below and Appendix B.

- Test family: F tests

- Statistical test: ANOVA
- Alpha error probability 0.05
- Effect size f^2: 0.40 - represent a relatively large effect size (Cohen, 1988)
- Power: 0.95 – normal convention.

8.3.4 Ethics

The Ethics Committee of the University of Southampton had approved this study, Ethics number ERGO/FPSE/30576. Thus, the user satisfaction survey proceeded under Ethics number ERGO/FPSE/30576, further details in Appendix B. This ethics number was distributed to participants before conducting the survey.

8.3.5 Pilot study

To make sure the survey operates efficiently, a pilot study will be undertaken. Around 10% (30 participants) of the anticipated number of users will participate in this pilot in order to improve the original questionnaire before conducting the real investigation. They were asked to read the participant information sheet and answered the questionnaires. The reliability of questionnaires was evaluated by Cronbach's alpha. This method was employed for calculating the internal consistency (Saunders et al., 1997). Each metric's calculated performance would be displayed in Table 8-1 as a reliability value. In Table 8-1, the Cronbach's alpha of each metric was shown from 0.731 to 0.918. The calculated result by Cronbach's alpha of the whole group was shown higher than 0.70. Therefore, it can be clearly seen that the designed questionnaire has a good consistency for conducting the real survey.

Table 8-1 List of metrics employed in the case study

Metric	Question	Cronbach's alpha
Functional experience	Do you feel satisfied with the displayed content in web page as usual? Do you feel satisfied with the displayed picture in web page as usual?	0.918

	▪ Do you feel satisfied with the type of displayed font language in web page as usual? ▪ Do you feel satisfied with links in web page to be able to use as usual? ▪ When I play video on the web browser, I feel that it works well. ▪ I feel that installed plugins can work well	
Hedonic experience	▪ I am satisfied with the performance of the web browser ▪ I feel happy because this web browser did not annoy me while surfing	0.731
Social experience	▪ I feel that I can use the web browser to communicate with other people smoothly ▪ I feel that using the web browser helps me to expand my social circle better	0.804
Pleasure	▪ I feel pleasure after using the web browser ▪ I feel gratified after using the web browser	0.792
Dominance	▪ Do you feel that you could pass the login website? ▪ Do you feel that you could logout from website?	0.853
Word of Mouth	▪ I would suggest this web browser installed countermeasure to other people for future use ▪ I would like to share a good experience for using this web browser installed countermeasure to other people	0.751

8.3.6 **Data Collection**

CP-Rmuti participants were invited face to face to participate in the survey with a researcher in their computer lab during 15 to 16 November 2017 as shown in Table 8-2. They had to read the participant information sheet before the survey took place. The printed versions were shown in Appendix B.

Table 8-2 Sample of CP-Rmuti participants

Characteristics	Number of participants	Percentage (%)
Gender		
Male	81	67.5%
Female	39	32.5%
Missing	-	-
Study Level		
1	16	13.3%
2	21	17.5%
3	60	50%
4	23	19.1%
Program		
Regular	120	100%
Special	-	-
Missing	-	-

8.3.7 Statistical analysis

Data analyses were performed using IBM SPSS statistics[96]. Given the number of fingerprint countermeasures used for testing the user browsing experience, the analysis of variance (ANOVA) is suitable for analysing in case of more than two means of the independent group. The mean values between groups will be compared through one way ANOVA followed by Tukey's HSD post hoc tests for multiple comparison with (p value < 0.05). The tested data need to conform to the assumptions of ANOVA, independent samples and normal distribution.

The null hypothesis of ANOVA (H_0) is that mean (average value) is the identical with all groups, while H_1= there are differences between two or more means. The hypotheses of interest in ANOVA are as follows:

$$H_0: \mu_1 = \mu_2 = \cdots = \mu_k$$

[96] https://www.ibm.com/analytics/data-science/predictive-analytics/spss-statistical-software

$$H_1: \ means\ are\ not\ equal$$

Usually, the significance level will be maintained at p≤ 0.05, it means that if p≤ 0.05, H0 will be rejected and H1 will be supported. Types of ANOVA consists of two types, one way ANOVA and two way ANOVA. Because this research uses only one independent variable (countermeasure), one way ANOVA will be used for analysis.

8.3.8 **Results**

Table 8-3 result of one way ANOVA

Descriptive				ANOVA				
Metrics	**countermeasures**	**Mean**	**Standard deviation**		**Mean square**	**df**	**F**	**p**
Functional experiences	FP-Block	3.0444	0.88293	Between groups	13.289	3	21.232	p<0.001
	Stop-fingerprinting	3.3667	0.91224					
	Fireglove	2.9778	0.69857	Within groups	0.626	116		
	FP-prevention	**4.4167**	**0.63540**					
Dominance	FP-Block	2.8500	1.04345	Between groups	21.847	3	21.246	p<0.001
	Stop-fingerprinting	3.3833	1.30439					
	Fireglove	2.5667	0.98902	Within groups	1.028	116		
	FP-prevention	**4.5000**	**0.58722**					
Social experience	FP-Block	2.4000	0.97733	Between groups	37.424	3	46.526	p<0.001
	Stop-fingerprinting	3.0167	1.05441					
	Fireglove	1.9333	0.94443	Within groups	0.804	116		
	FP-prevention	**4.5000**	**0.5855**					
Hedonic experience	FP-Block	2.3833	0.94398	Between groups	31.997	3	39.842	p<0.001
	Stop-fingerprinting	2.7500	1.04840					
	Fireglove	2.1500	0.88230	Within groups	0.803	116		
	FP-prevention	**4.4333**	**0.66609**					
Pleasure	FP-Block	2.1000	0.88474	Between groups	36.447	3	50.785	p<0.001
	Stop-fingerprinting	3.1167	1.05604					
	Fireglove	2.1167	0.75067	Within groups	0.718	116		
	FP-prevention	**4.4333**	**0.63968**					
Word of Mouth	FP-Block	2.3667	1.06620	Between groups	34.103	3	35.003	p<0.001
	Stop-fingerprinting	2.7167	1.09610					
	Fireglove	2.1333	1.09019	Within groups	0.974	116		
	FP-prevention	**4.4833**	**0.60861**					

A one way between subjects ANOVA was conducted to compare the effect of user browser experience on the web browser with fingerprint countermeasures through six metrics.

Considering functional experience, there was statistically significant effect of effect of user browser experience on the web browser with fingerprint countermeasures at $p<0.001$ level for the four fingerprint countermeasures [$F(3,116) =21.232$, $p<0.001$].

Considering dominance, there was statistically significant effect of effect of user browser experience on the web browser with fingerprint countermeasures at $p<0.001$ level for the four fingerprint countermeasures [$F(3,116) =21.246$, $p<0.001$].

Considering social experience, there was statistically significant effect of effect of user browser experience on the web browser with fingerprint countermeasures at $p<0.001$ level for the four fingerprint countermeasures [$F(3,116) =46.526$, $p<0.001$].

Considering hedonic experience, there was statistically significant effect of effect of user browser experience on the web browser with fingerprint countermeasures at $p<0.001$ level for the four fingerprint countermeasures [$F(3,116) =39.842$, $p<0.001$].

Considering pleasure, there was statistically significant effect of effect of user browser experience on the web browser with fingerprint countermeasures at $p<0.001$ level for the four fingerprint countermeasures [$F(3,116) =50.785$, $p<0.001$].

Finally, considering word of mouth, there was statistically significant effect of effect of user browser experience on the web browser with fingerprint countermeasures at $p<0.001$ level for the four fingerprint countermeasures [$F(3,116) =35.003$, $p<0.001$].

To sum up in Table 8-3, the F statistic is relatively high and p value is less than 0.01 for all metrics. Therefore, the research rejects the null hypothesis (H_0) and accepts the alternative hypothesis (H_1). It means that there are statistically significant differences between groups as shown by one-way ANOVA (additional results provided in Appendix C).

Results of multiple comparison

In previous results in Table 8-3, the research can learn the significance value between groups generated by using one-way ANOVA. It can be summarized that there are

significant differences between groups. However, the results did not show which groups show the most significant effects. Therefore, the research selects a Tukey post hoc test in order to find out which groups show the most significant effects further. The Tukey post hoc test compares FP-prevention with FP-Block, Stop-fingerprinting, and Fireglove and shows results of comparison as shown in Table 8-4.

To sum up, a Tukey post hoc test had been conducted to compare differences between groups. It reveals that FP-prevention shows scores statistically significantly higher than Fireglove, FP-prevention, and Stop fingerprinting at all metrics after taking the Functional experiences (4.41 ± 0.63540, $p < 0.05$), the Dominance (4.50 ± 0.58722, $p < 0.05$), Social experience (4.50 ± 0.50855, $p < 0.05$), the Hedonic experience (4.43 ± 0.66609, $p < 0.05$), Pleasure (4.43 ± 0.63968, $p < 0.05$) and Word of Mouth (4.48 ± 0.60861, $p < 0.05$). Therefore, it can be concluded that FP-prevention shows a statistically significant improvement compared to Fireglove, FP-prevention, and Stop fingerprinting as shown in Figure 8-2 (additional results provided in Appendix C).

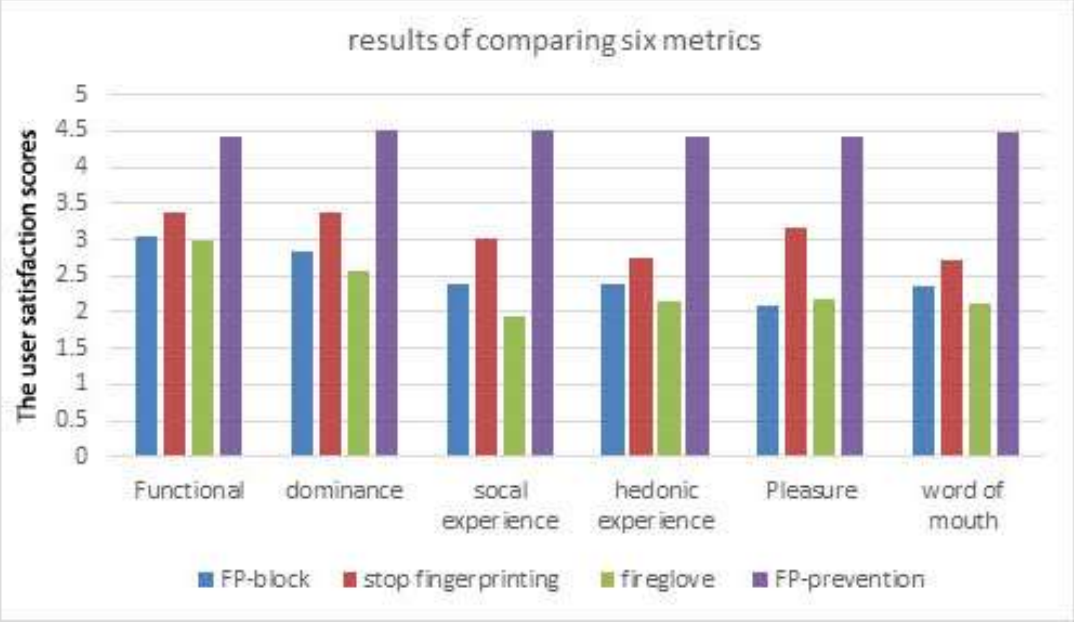


Figure 8-2 results of user experience with all metrics

Table 8-4 the partial results of multiple comparisons all metrics through Tuekey Post hoc tests

		Functional Experience		Dominance		Social Experience		Hedonic Experience		Pleasure		Word of Mouth	
(I) countermeasure	(J) countermeasure	Mean Difference (I-J)	Sig.	Mean Difference (I-J)	Sig.	Mean Difference (I-J)	Sig.	Mean Difference (I-J)	Sig.	Mean Difference (I-J)	Sig.	Mean Difference (I-J)	Sig.
FP-Block	stop_fingerprinting	-.32222	.395	-.53333	.181	-.61667*	.043	-.36667	.391	-1.01667*	< 0.001	-.35000	.519
	Fireglove	.06667	.988	.28333	.701	.46667	.188	.23333	.745	-.01667	1.000	.23333	.797
	FP_prevention	-1.37222*	< 0.001	-1.65000*	< 0.001	-2.10000*	< 0.001	-2.05000*	< 0.001	-2.33333*	< 0.001	-2.11667*	< 0.001
stop_fingerprinting	FP-Block	.32222	.395	.53333	.181	.61667*	.043	.36667	.391	1.01667*	< 0.001	.35000	.519
	Fireglove	.38889	.232	.81667*	.012	1.08333*	< 0.001	.60000	.052	1.00000*	< 0.001	.58333	.107
	FP_prevention	-1.05000*	< 0.001	-1.11667*	< 0.001	-1.48333*	< 0.001	-1.68333*	< 0.001	-1.31667*	< 0.001	-1.76667*	< 0.001
Fireglove	FP-Block	-.06667	.988	-.28333	.701	-.46667	.188	-.23333	.745	.01667	1.000	-.23333	.797
	stop_fingerprinting	-.38889	.232	-.81667*	.012	-1.08333*	< 0.001	-.60000	.052	-1.00000*	< 0.001	-.58333	.107
	FP_prevention	-1.43889*	< 0.001	-1.93333*	< 0.001	-2.56667*	< 0.001	-2.28333*	< 0.001	-2.31667*	< 0.001	-2.35000*	< 0.001
FP_prevention	FP-Block	1.37222*	< 0.001	1.65000*	< 0.001	2.10000*	< 0.001	2.05000*	< 0.001	2.33333*	< 0.001	2.11667*	< 0.001
	stop_fingerprinting	1.05000*	< 0.001	1.11667*	< 0.001	1.48333*	< 0.001	1.68333*	< 0.001	1.31667*	< 0.001	1.76667*	< 0.001
	Fireglove	1.43889*	< 0.001	1.93333*	< 0.001	2.56667*	< 0.001	2.28333*	< 0.001	2.31667*	< 0.001	2.35000*	< 0.001

8.4 Summary

Details of methodologies about the case study conducted are described in this chapter. In order to measure the user browsing experience, the student satisfaction survey was undertaken at the department of computer technology of Rajamangala University of Technology Isan Surin campus (CP-Rmuti). The data collected from the student satisfaction survey consisted of six metrics for measuring the user browsing experience. The student satisfaction survey was analysed through one-way ANOVA followed by Tukey's HSD post hoc tests for multiple comparisons. The result of ANOVA revealed that FP-prevention showed statistically significant improvement, compared with other countermeasures. Stop fingerprinting is ranked as a second highest score of user satisfaction followed by FP-block and Fireglove respectively.

Chapter 9 Conclusion and Future work

This chapter starts with the overview of this research and then discussion about answering the research question if this thesis in Section 9.2. Next, Section 9.3 explains the contribution of this research after completion. The discussion about the future work in Section 9.4 of this research will be divided into three main parts, namely development of fingerprinting detection, sustainable solution and future of fingerprinting technique, followed by the summary of the thesis.

9.1 Overview of this research

Inhibiting fingerprint tracking is highly important for users who require to maintain their privacy on the Internet. Currently, users are able to find available countermeasures through the Internet against fingerprint tracking described in Chapter 3, with different techniques of fingerprint prevention to assist the Internet user. However, although these techniques are able to prevent fingerprint tracking, they have the significant disadvantage of reducing web browser functionality with a corresponding loss of user experience. It is clear that almost all countermeasures focus only on the fingerprinting prevention without paying sufficient attention to the problems with the user experience.

For this reason, this research proposes a countermeasure to prevent the fingerprinting tracking with the minimum impact on a web browser. The new countermeasure, called FP-prevention, has enhanced a basic concept from Torres et al. (2015), who used the technique of changing attributes on the web browser by considering the relationship of attributes. As mentioned in the empirical study in Chapter 5, the technique of Torres et al. (2015) still has limitations, especially in terms of introducing problems with the web browser. In order to address these problems, this research has introduced a new technique to solve the existing problems by considering the consequences of changing attributes along with regarding the relationship between attributes.

The key innovation with FP-prevention is to consider the sequence of changing attributes and selecting an appropriate sequence for each attribute to mitigate the problems with user experience. The ability to link sessions that is the main problem with fingerprinting tracking would be eliminated by changing the user identification every time they visit the web server. Alteration of attributes with FP-prevention would produce minimum side-effects to the web browser and mitigate problems of a web browser being identified.

Using FP-prevention is expected to assist users to avoid fingerprint tracking as well as producing the minimum impact on the web browser. Any users using FP-prevention should be able to use their web browsers to navigate websites as usual. The results of fingerprinting prevention by using FP-prevention were validated through the experiments in Chapter 7, while the results of side-effects using FP-prevention were verified through the survey in Chapter 8.

9.2 Answering the research question

The research question was addressed by use of the research study. The research question is the key contribution of this thesis:

How can randomized attributes be introduced into a web browser in order to prevent browser fingerprinting with a minimal impact on the browsing experience?

In order to answer the research question, the preliminary experiment is intended to find out which methods have a strong impact on the web browser, which changed attributes create side effects on the web browser and why the web browser with fingerprint countermeasure installed is easier to fingerprint. To address this research question, several countermeasures that can be used for inhibiting the fingerprint tracking were selected and tested based on previous comparative studies.

The next phase focuses on designing a methodology for developing a new fingerprint countermeasure. The additional details regarding the development of FP-prevention can be found in Chapter 6.

The final phase is related to the evaluation of FP-prevention analysis by measuring the performance of a web browser, the effectiveness of fingerprinting prevention, information paradox and using survey research method as a case study. The details of the evaluation methodology and results are described in Chapter 7 and Chapter 8 respectively.

In conclusion, the outcome of fingerprinting prevention through FP-prevention is evaluated by several approaches through the comparison with other countermeasures. The analytical results show that the quality of FP-prevention is regarded as an acceptable outcome.

9.3 Contribution

The whole process of this research, from the empirical and actual experiments conducted, has enhanced learning and extended knowledge in the field of browser fingerprinting. The main contributions of this research can be summarized as follows:

- **Knowing about nature of changed attributes**

After learning a basic process of creating the browser fingerprinting technique in Chapter 5.1, the results of observing fingerprinted attributes revealed that a web browser version is frequently updated, while other attributes such as fonts or plugin, are not frequently changed. In addition, updating the version of a web browser every time would change some value of screen objects such as availableLeft, availTop, availHeight and availWidth as well. Knowing how many times selected attributes are changed per month assists the fingerprinters to improve their fingerprint script according to the nature of their selected attributes. In addition, the research established that characteristics of attribute data used for randomization directly affects the user browser experience. For example, the latest version of User-agent shows insignificant side effects, while old versions of User-agent showed significant side effects. Furthermore, some attributes cannot be changed arbitrarily because they cause side effects on the web browser (e.g., Language or Timezone). Changing any attributes must realize the scope of side effects before changing them.

- **Enhancing the current fingerprint countermeasure**

FP-prevention is designed to assist users because none of the current browser vendors provides a feature to inhibit the fingerprint tracking. By providing continually changing values, FP-prevention is able to break the stability of fingerprinting and ensure that fingerprinters cannot track a user, as shown in Chapter 7.

The main drawback of the current countermeasures is to change attributes without understanding their characteristics. FP-prevention is intended to address this problem by introducing a new way to change attributes. The selected attributes will be changed by the FP-prevention model which will mitigate the problems of the user browsing experience and the information paradox. The insight behind FP-prevention derives from the results of empirical experiments which reveal that understanding the characteristics of attributes is extremely important to mitigate possible problems with the user browsing experience. The changing of attributes should use a model to regulate workflow of changed attributes, rather than changing them arbitrarily.

In terms of advantages of FP-prevention, FP-prevention yields the highest score of all metrics of user satisfaction through a survey. It can be clearly seen that using FP-prevention shows negligible side effects on the user experience, compared with other countermeasures. As for web browser performance, a web browser with FP-prevention installed and one without it are measured on the execution time by three JavaScript benchmarks. The obtained results show that installing FP-prevention has no effect on web browser performance, judging from execution time from two of three JavaScript benchmarks. Therefore, it can be clearly seen that a user installing FP-prevention can use their web browsers as normal. As for the information paradox, the results of seven metrics tested clearly show that a web browser used with FP-prevention yields the lowest score with all metrics related to the problem of information paradox. Thus, it can be clearly seen that using FP-prevention did not exacerbate problems of information paradox, compared with other countermeasures.

In contrast, FP-prevention shows poor performance in terms of effectiveness of fingerprinting prevention. The results of observing fingerprint ID multiple times provided by three fingerprinting websites suggest that FP-prevention is ranked as the third most effective countermeasure. In addition, sometimes FP-prevention's code inadvertently

breaks the execution of the fingerprinting code resulting in providing some null values to the fingerprinter. This problem results in the fingerprinting prevention being less effective compared with FP-block and Stop-Fingerprinting. Furthermore, a disadvantage of FP-prevention is it needs to frequently update the latest version of User-agent used for randomization. Regardless of updating the latest version of User-agent, it directly affects the user browser experience.

9.4 Future work

The emergence of web technologies (e.g., HTML5, JavaScript, Plugins or CSS3) has assisted users to interact with the web conveniently on an unprecedented scale. The diversity of APIs, variables and functions provided by web technologies have allowed the browser fingerprinting technique to become viable. Even though browser fingerprinting appears to be a simple technique, solving this problem is not as straightforward as it might appear at first. Relying on only a single technique to cope with the problem of fingerprinting is definitely not sufficient. Every version of a modern web browser update, every new function or ability introduced by W3C[97] and every new web technology emerging will enhance the ability of the fingerprinters to identify users in the future. This section will discuss the future of fingerprinting techniques that might affect both developers and normal Internet users.

9.4.1 **Development of fingerprinting detection**

Occasionally, users may need to recognize which websites attempt to track them with the browser fingerprinting technique. Even though it is extremely difficult to identify whether the execution of JavaScript code loading is a normal script or a fingerprinting code, fingerprinting code detection should be developed further because this is an essential prerequisite to inhibit the execution of fingerprint code. In order to accomplish this, understanding in-depth how JavaScript is executed is extremely important. Program analysis(such as static analysis (Orr et al., 2012) and dynamic analysis (Tran et al., 2012)), is relatively common in the area of web security. Unfortunately, developing these techniques

[97] https://www.w3.org/

is not sufficient to identify fingerprinting code. This section discusses two aspects, namely indicators of being fingerprinted and the obstacles of detecting the fingerprinting code, in order to guide an interested person to study fingerprinting further.

9.4.1.1 Indicators of being fingerprinted

The author has listed possible indicators of being fingerprinted through the browser fingerprinting technique.

- **Using a hash function:** The last process of browser fingerprinting technique is to create the fingerprinting ID by using a hash function and then send this hash back to the fingerprinting server. Detecting use of a hash function in executed script is likely to imply that the executed script is attempting to fingerprint a user.
- **Attempting to access the specific functions:** This is a straightforward method to detect the fingerprinting code. Even though fingerprinters can use any attributes on the user computer through a browser for fingerprinting, there are a few distinctive attributes more likely to be used by fingerprinters according to the literature review in Chapter 2 such as platform, User-agent or version of a web browser. Accessing the specific functions at the same time is likely to imply that the web browser is being fingerprinted.
- **Attempting to gather a large quantity of user's device information:** Observing whether an executed script attempts to access attributes related to the user's information or not and how many attributes are accessed. A number of attributes related to the user's information can imply a sizeable probability of being fingerprinted through the browser fingerprinting technique.
- **Accessing the same object or same function many times:** calling any functions or the same object many times can be considered to be a sign of fingerprinting. For example, if a user is fingerprinted by plugins fingerprinting, the navigator.plugins will be called many times in order to gather the list of plugins.

Finally, all suggestions above must send this information (Users' information and the hash ID) back to the remote server where it could be observed. However, the author considers that all of the indicators as suggested above contribute to a determination of the suspiciousness level, but this may not be accurate enough for detecting the fingerprinting

code. Planning to use the information flow analysis to detect the fingerprinting attempts is one avenue to pursue in future.

9.4.1.2 The obstacles for detecting the fingerprinting code

Even though many indicators of being fingerprinted had been suggested, today an executed script on a web browser is more complicated and it can be extremely difficult to identify any fingerprint code.

- **Obfuscated code**: Many fingerprinters have many methods to obfuscate the fingerprinting code. A famous method used for obfuscating is "*minimization*". This method is extremely useful to minimize the amount of data that needs to be transferred by removing comments and whitespace such as JSmin[98]. In addition, using this method can also modify variables (e.g., rename to be short). Unsurprisingly, many developers employ obfuscation in order to protect or hide their code for various reasons.

  In order to solve the problem of obfuscation of code, reverse-engineering is required which takes time to pinpoint how the fingerprinting code works.

- **Separated task to other sites:** Rather than using only minimization, this method supplements the level of obfuscating fingerprinting code, making it so difficult that nobody can understand at all how the fingerprinting code performs. This method will divide a single fingerprinting script into many scripts and then place the divided fingerprinting script to different sites rather than placing one fingerprinting script in one site as usual. This method poses technical challenges for researchers on how to analyse fingerprinting code in the future.

  Instead of examining only the single script as usual, addressing this problem requires examining at the level of a website loading the script along with examining the single script.

[98] http://www.crockford.com/javascript/jsmin.html

9.4.2 Sustainable Solution

The main root of problems of being tracked by the browser fingerprint technique results from obtaining a user's basic information through interaction between a web server and a web browser. Connection between the web browser and a web server must pass through many layers to successfully complete the process of connection. Every interacted layer can provide a user's basic information to the web server that can be used to fingerprint. Nikiforakis et al. (2013) had classified the possibility of being fingerprinted in an interesting way with a taxonomy of possible attributes used by fingerprinters.

Table 9-1 Taxonomy of all attributes used by fingerprinters

Fingerprinting Category	Example of used attributes
Browser customization	Plugins, MimeType, ActiveX, Google gear Detections
Browser level user configurations	Cookies, TimeZone, Date time, Browser language or Do not Track
Browser family & Version	User_agent, Accept-Header, Math constant or AJAX implementation
Operating system & Application	Font, operation system, OS kernel version or Window Registry.
Hardware & Network	Screen resolution, IP address or TCP/IP parameter

Considering Table 9-1, it can be clearly seen that a web server can learn a user's basic information with all categories as required. This does not include the difference of functionalities in each web browser (Nikiforakis et al., 2013), difference of ordering JavaScript attributes (Laperdrix et al., 2017) and a difference of responding to a JavaScript engine of each web browser (Mulazzani et al., 2013) that probably are able to fingerprint a user. Therefore, the idea that the browser fingerprinting technique is difficult to detect, to prevent and to hide is still the absolute truth because users and developers cannot even guess what kind of fingerprinting technique is being used to fingerprint them.

As long as many modern web browsers still did not pay attention to the problem of browser fingerprinting technique, this problem still exists. The problem of the browser

fingerprinting technique is unable to be fixed with a simple patch but it must consider the overall picture. The root of problems is that a web server can learn the user's data with different layers through the web browser while interacting (Table 9-1). In the long run, the sustainable solution should originate from a modern web browser with a genuine awareness of the problem of browser fingerprinting technique and that is willing to provide a new private mode to the user. In order to solve the whole problem of fingerprinting technique, the future private mode should be provided as follows.

- **Using standardized API calls**

The author strongly agrees with Boda et al. (2012a) that using common API calls to disclose a web server is an inevitable part. Using common APIs calls will reduce the difference of attribute values resulting in the fingerprinting.

- **Reducing redundant attributes**

Currently, there are many attributes used in a web browser. A large number of attributes facilitates the browser fingerprinting technique to operate smoothly. Therefore, redundant attributes should be reduced in order to sidestep unnecessary increase to the fingerprint surface.

- **Specifying ordering of attributes differences**

The differences of ordering of attribute enumeration of browser objects allows a web browser to be fingerprinted (Laperdrix et al., 2017). For example, the order of enumeration of navigator and screen objects is different between browser families and versions of each web browser. This allows fingerprinters to use this weakness for fingerprinting. This problem should be solved by specifying return values that will not open a soft target for fingerprinters.

- **Reducing fingerprinting surface**

Eliminating the source of entropy and accessible attributes used for fingerprinting are extremely important. The attributes that are likely to be used for fingerprinting (e.g., fonts) will be marked and then the number of these attributes should be decreased in order to reduce the fingerprinting surface.

9.4.3 Future of fingerprinting technique

Even though the web cookie is a well-established technique, it remains in widespread current use for various reasons. One of the main reasons is the accuracy of web cookies. The accuracy of web cookies derives from the good stability of identifying a user more effectively than the browser fingerprinting technique. The concept of fingerprinting is more effective and accurate when applied to the human finger because the data obtained for a human fingerprint is unchangeable or changes only very gradually. When this technique is implemented with the user's computer, the stability and diversity are a major problem for the browser fingerprinting technique because some attributes on the user computer are changeable. According to the latest result of Gómez-Boix et al. (2018), they analysed 2,067,942 browser fingerprint through top 15 French websites. They established that browser fingerprint technique did not give an effective mechanism to identify a user uniquely. This is the main reason why the browser fingerprinting technique cannot replace web cookies as the main technique to track a user.

In terms of security, the browser fingerprinting technique is similar to the security script. Many secure businesses still use this technique to identify the genuine user with their web authentication. In future, this technique will be used more and more, especially by the business sector, in order to battle against user impersonation.

In respect to the scale of identifying a user, another question has been raised about the scale of identification if the number of users is extremely large in practice. For example, suppose a fingerprint site needs to deal with millions of users a day. Dealing with millions of unique fingerprints a day has unpredictable consequences on how to manage them effectively, including the issue of dealing with duplicate fingerprints. For instance, learning updates of a large number of users' computers over time is a huge task. Currently, the Panopticlick[99] project collects only 1,524,157 users visiting per year, not a million users visiting every day.

[99] https://panopticlick.eff.org/

9.5 Summary of Thesis

The main motivation of this thesis is the observed deficiency in protecting the user privacy against fingerprint tracking. The majority of fingerprint countermeasures that are available online cannot prevent fingerprinting without having a significant impact on the web browsing experience. The main concept set out in this thesis is to address the problem of fingerprint tracking along with reducing the side-effects to the web browser. The improvement of FP-prevention yields the following outcomes:

1) Thorough analysis of empirical studies of fingerprint countermeasures
2) Thorough analysis of execution time by using three JavaScript performance benchmark suites for analyse.
3) Thorough analysis of effectiveness of fingerprinting prevention by comparing with other countermeasures.
4) Thorough analysis of information paradox compared with other countermeasures.
5) Thorough analysis of user experience through a PAD model for analysis.

It is expected that the methodology of this research can be used as guidance for researchers and practitioners in finding and improving a way to prevent the fingerprint tracking in the future. Furthermore, it is also fully expected that the future studies will provide the correct answer to the argument concerning how to prevent the fingerprint tracking with the minimum impact to the web browser. In addition, it is expected that the FP-prevention introduced in this research can pave the way for future development of fingerprint countermeasures.

Lastly, the author has full confidence in the process of developing and evaluating the FP-prevention technique because of the research methods adopted. FP-prevention can currently be used as an effective fingerprinting countermeasure while having only minimum impact on the web browsing experience.

Appendix A Details of attributes to study

The main purpose of the study is to observe suspect attributes which are likely to be used for fingerprinting as much as possible. In this study, many attributes of JavaScript object were selected from the literature review and fingerprinting code as follows.

Attributes associated with navigator object consisted of 15 attributes. These attributes provide a user's basic information to a server, classified as the noticeable attributes. In terms of plugin and mimeType, these attributes are relatively popular among fingerprinters because of their high entropy. This research chooses four attributes associated with plugin and mimeType to observe a change of attributes. Finally, 11 attributes related to the screen object will be observed in this study because these attributes are relatively constant, rarely changing. For this reason, many fingerprinters may choose these attributes for fingerprinting.

Table A-1 list attributes associated with navigator object

Navigator (15 attributes)	Panopticlick	Bluecava	Iovation	ThreatMetrix	AddThis	FingerPrintJS	FP-detective
User-agent	✓	✓	✓	✓	✓	✓	✓
appCodeName						✓	✓
appVersion						✓	✓
doNotTrack	✓				✓	✓	
product						✓	✓
productSub						✓	✓
cookieEnabled	✓	✓	✓	✓	✓	✓	✓
vendor						✓	✓
vendorSub						✓	✓
online						✓	✓
platform	✓	✓	✓	✓	✓	✓	✓
language	✓	✓	✓	✓	✓	✓	✓
languages						✓	✓
mimeTypes					✓		
JavaEnabled	✓				✓	✓	✓

Table A-2 list attributes associated with plugin

Navigator.plugins (4 attributes)	**Panopticlick**	**Bluecava**	**Iovation**	**ThreatMetrix**	**AddThis**	**FingerPrintJS**	**FP-Detective**
name	✓	✓	✓	✓	✓	✓	✓
filename	✓	✓	✓	✓	✓	✓	✓
description	✓	✓	✓	✓	✓	✓	✓
length	✓	✓	✓	✓	✓	✓	✓

Table A-3 list attributes associated with mimeType

Navigator.mimeTypes (4 attributes)	**Panopticlick**	**Bluecava**	**Iovation**	**ThreatMetrix**	**AddThis**	**FingerPrintJS**	**FP-Detective**
enablePlugin				✓		✓	✓
description				✓		✓	✓
suffix				✓		✓	✓
type				✓		✓	✓

Table A-4 list attributes associated with screen object

Screen (11 attributes)	**Panopticlick**	**Bluecava**	**Iovation**	**ThreatMetrix**	**AddThis**	**FingerPrintJS**	**FP-Detective**
horizontalDPI							✓
verticalDPI							✓
availLeft							✓
availTop							✓
availHeight							✓
availWidth							✓
colorDepth	✓					✓	✓
pixelDepth							✓
width	✓					✓	✓
Height	✓					✓	✓
bufferDepth							✓

Appendix B Student satisfaction survey material

B.1 Student Satisfaction Survey

Department of computer technology, Faculty of Agriculture and Technology,
Rajamangala University of Technology Isan, Surin campus.

The survey would ask students about level of satisfaction with using the fingerprint countermeasure installed on their web browsers.

Section I Personal Information

Direction: Please tick the circle that the best corresponds to your answer for each question below

1. Gender ○ Male ○ Female
2. Study Level ○ 1 ○ 2 ○3 ○ 4 ○ other (specify)

Section II is a level of user satisfaction with using the fingerprint countermeasure installed on their web browsers at Department of computer technology, Faculty of Agriculture and Technology, Rajamangala University of Technology Isan, Surin campus.

Direction: Considering your experience with using the fingerprint countermeasure installed on web browser, please pick your level of satisfaction with the fingerprint countermeasure installed on your web browser by ticking the response corresponding to your opinion by using the following scales:

Score	Level of Satisfaction
5	Very Satisfied
4	Satisfied
3	Neutral
2	Dissatisfied
1	Very Dissatisfied

Functional experience	**Level of Satisfaction**				
	5	4	3	2	1
Do you feel satisfied with the displayed content in web page as usual?					
Do you feel satisfied with the displayed picture in web page as usual?					
Do you feel satisfied with the type of displayed font language in web page as usual?					
Do you feel satisfied with links in web page to be able to use as usual?					
When I play video on the web browser, I feel that it works well					
I feel that installed plugins can work well					

Social experience	**Level of Satisfaction**				
	5	4	3	2	1
I feel that I can use the web browser to communicate other people smoothly					
I feel that using a web browser help me to expand my social circle better					

Dominance	**Level of Satisfaction**				
	5	4	3	2	1
Do you feel that you could pass the login website?					
Do you feel that you could logout from website?					

Hedonic experience	**Level of Satisfaction**				
	5	4	3	2	1
I am satisfied with the performance of a web browser					
I feel happy because this web browser did not annoy me while surfing					

Pleasure	**Level of Satisfaction**				
	5	4	3	2	1
I feel pleasure after using a web browser					
I feel gratified after using a web browser					

Word of Mouth	**Level of Satisfaction**				
	5	4	3	2	1
I would suggest this web browser installed countermeasure to other people for future use					
I would like to share a good experience for using this web browser installed countermeasure to other people					

---------- Thank you for taking the time to complete this questionnaire ----------

Appendix B

G2. Language Consistency

Brief of language consistency translation test of student satisfaction

Question/Instruction (English)	**Question/Instruction (Thai)**	**Language consistency score**			**Total Score**	**Comment**
		Expert 1	Expert 2	Expert 3		
The survey would ask students about a level of satisfaction using a fingerprint countermeasure installed on a web browser.	แบบประเมินนี้จะถามความคิดเห็นของนิสิตเกี่ยวกับระดับความพึงพอใจในการใช้ต้านองกันการติดตามด้วยเทคนิคลายนิ้วมือซึ่งถูกติดตั้งในเว็บบรเซอร์	+1	+1	0	2	Replace “นิสิต” with “นักศึกษา”
Section I Personal Information	ส่วนที่ 1 ข้อมูลพื้นฐานเกี่ยวกับผู้ตอบแบบสอบถาม	+1	+1	+1	3	
Direction: Please tick (✓) in table to express how do you feel with web browser being used.	คำชี้แจง: กรุณาทำเครื่องหมายถูก (✓) ในตารางเพื่อแสดงความรู้สึกสิ่งที่นักศึกษาคิดต่อเว็บบรเซอร์ที่กำลังใช้	+1	+1	+1	3	
Gender ○ Male ○ Female	เพศ ○ ชาย ○ หญิง	+1	+1	+1	3	

Level of education	ระดับการศึกษา	+1	0	1+	2	Replace"ระดับการศึกษา" with "ชั้นปีที่ทำการศึกษา"
Section II is measuring a level of user satisfaction with using the fingerprint countermeasure installed on web browsers at Department of computer technology, Faculty of Agriculture and Technology, Rajamangala University of Technology Isan, Surin campus.	ส่วนที่ 2 เป็นระดับความพึงพอใจของผู้ใช้กับตัวที่ทำการป้องกันการติดตามด้วยเทคนิคลายนิ้วมือซึ่งถูกติดตั้งในเว็บบราวเซอร์ ที่สาขาเทคโนโลยีคอมพิวเตอร์ คณะเกษตรศาสตร์และเทคโนโลยี มหาวิทยาลัยราชมงคลอีสาน วิทยาเขตสุรินทร์	1+	-1	+1	1	Suggest to use "วัดระดับความพึงพอใจ"
Direction: Considering your experience with using the fingerprint countermeasure installed on web browser, please pick your level of satisfaction with the fingerprint countermeasure installed on your web browser by ticking (✓) the response	คำชี้แจง: กรุณาระบุความพึงพอใจของคุณต่อตัวที่การป้องกันการติดตามด้วยเทคนิคลายนิ้วมือซึ่งถูกติดตั้งในเว็บบราวเซอร์ โดยการทำเครื่องหมายถูก (✓) ในช่องตัวเลขที่ตรงกับความคิดเห็นของนักศึกษา โดยใช้ระดับความคิดเห็นตามช่วงดังต่อไปนี้	0	+1	+1	2	Suggest to use "ต่อตัวโปรแกรมที่ทำการป้องกัน"

Appendix B

corresponding to your opinion by using the following scales:						
Level of Satisfaction Very Satisfied Satisfied Neutral Dissatisfied Very Dissatisfied	ระดับความพึงพอใจ พึงพอใจมาก พึงพอใจ ปานกลาง ไม่พึงพอใจ ไม่พึงพอใจมาก	+1	+1	+1	3	
Functional experience Do you feel satisfied with the displayed content in web page as usual?	**ประสบการณ์จากฟังก์ชั่นการทำงาน** คุณรู้สึกพอใจว่าเนื้อหาที่ถูกแสดงในหน้าเว็บเพจนั้นแสดงได้ถูกต้องปกติดี	0	+1	+1	2	Suggest to use “หน้าเว็บเพจนั้นแสดงได้ถูกต้องตามปกติดี“
Do you feel satisfied with the displayed picture in web page as usual?	คุณรู้สึกพอใจกับรูปภาพที่แสดงอยู่ในเว็บบราวเซอร์นั้นดูปกติดี	0	0	+1	1	Suggest to use “รู้สึกพอใจกับรูปภาพที่ถูกแสดง“
Do you feel satisfied with the type of displayed font language in web page as usual?	คุณรู้สึกพอใจว่ารูปแบบภาษาของฟอนต์ที่ถูกแสดงบนเว็บบราวเซอร์นั้นดูปกติดี	0	1+	+1	2	Replace “ดูปกติ” with “ไม่มีปัญหาในการอ่าน”
Do you feel satisfied with links in web page to be able to use as usual?	คุณรู้สึกพอใจกับลิงค์ต่างๆ ในหน้าเว็บเพจว่าสามารถใช้งานได้ปกติดี	+1	+1	+1	3	

When I play video on the web browser, I feel that it works well.	เมื่อฉันเล่นไฟล์วีดีโอในเว็บบราวเซอร์ ฉันรู้สึกว่ามันทำงานได้ปกติดี	0	+1	+1	2	Replace "ได้ปกติดี" with "เป็นปกติดี"
I feel that installed plugins can work well	ฉันรู้สึกว่าปลั๊กอินต่างๆ ที่ถูกตั้งตั้งสามารถใช้งานได้ปกติได้ดี	+1	+1	0	2	
Hedonic experience	**ความรู้สึกชอบ**	1+	0	+1	2	
I am satisfied with the performance of a web browser	ฉันรู้สึกพอใจกับการทำงานของเว็บบราวเซอร์					
I feel happy because this web browser did not annoy me while surfing	ฉันรู้สึกพอใจเพราะว่าเว็บบราวเซอร์ที่ถูกติดตั้งนี้ไม่ทำให้คุณรู้สึกว่ามันรำคาญเลยในขณะที่ใช้งาน	+1	+1	+1	3	
Social experience	**ประสบการณ์ทางการติดต่อกับผู้อื่น**	+1	+1	+1	3	
I feel that I can use the web browser to communicate other people smoothly	ฉันรู้สึกว่าฉันสามารถใช้เว็บบราวเซอร์ติดต่อกับคนอื่นได้ดี					
I feel that using a web browser help me to expand my social circle better	ฉันรู้สึกว่าการใช้เว็บเบราเซอร์ช่วยให้ฉันสามารถขยายการติดต่อในสังคมของฉันได้ดี	+1	+0	+1	2	

Appendix B

Pleasure I feel pleasure after using a web browser	ความสุข ฉันรู้สึกพอใจหลังจากคุณเล่นมัน	0	+1	+1	2	Replace "รู้สึกพอใจ" with "รู้สึกดี "
I feel gratified after using a web browser	ฉันรู้สึกชอบใจหลังจากใช้เว็บเบราเซอร์	1	1	0	2	
Dominance Do you feel that you could pass the login website?	การสามารถควบคุมได้ คุณรู้สึกว่าคุณสามารถล็อกอินเข้าสู่เว็บไซต์ได้ดีหรือไหม	+1	0	+1	2	
Do you feel that you could logout from website?	คุณรู้สึกว่าคุณสามารถล็อกเอาท์ออกจากเว็บไซต์ได้ดีหรือไหม	+1	-1	+1	1	Replace "could logout" with "can logout"
Word of Mouth I would suggest this web browser installed countermeasure to other people for future use	แบบปากต่อปาก ฉันอยากแนะนำเว็บบราวเซอร์นี้ให้คนอื่นใช้ในอนาคต	+1	0	+1	2	Replace "เว็บบราวเซอร์" with "เว็บบราวเซอร์ที่ลงตัวป้องกัน"
I would like to share a good experience for using this web browser installed countermeasure to other people	ฉันอยากจะแชร์ความรู้สึกดีๆ ในการใช้เว็บบราวเซอร์ที่ลงตัวป้องกันนี้กับผู้อื่น	+1	0	+1	2	Replace "รู้สึกดี" with "รู้สึกที่ดี"

B.2 Participant Information Sheet

Study Title: A study of side-effects of using fingerprint countermeasures installed on a web browser.

Investigator: Sakchan Luangmaneerote
Ethics number: **ERGO**/FPSE/30576

Please read this information carefully before deciding to take part in this research. If you are happy to participate you will be asked to sign a consent form.

What is the research about?
This research is attempting to assist a computer user to avoid the tracking with the fingerprinting technique while navigating the websites. In order to avoid tracking, each fingerprint countermeasure would be installed on the web browser, and then users would use their web browser to observe side-effect of using. For side-effects, it means that a web browser cannot run smoothly, malfunction of plugins installed on the web browser, problems of functionalities and display of web page. The research would like to know whether the fingerprint countermeasures being tested has side-effects to the web browser or not.

Why have I been chosen?
At this stage, this study is trying to collect information about side-effects of each fingerprint countermeasure while using the web browser. The research requires a user who usually uses the Internet to give opinions about a web browser being used. Thus, skills in the use of computer and use the Internet is essential to assess the fingerprint countermeasure installed in a web browser.

What will happen to me if I take part?
If you are a computer user, you will be invited to use a web browser installing a fingerprint countermeasure. You need to reflect your opinions about running a web browser while navigating the websites. It will take 10-20 minutes to complete questionnaires.

Are there any benefits in my taking part?
By taking part, you have the opportunity to help us development the new fingerprint countermeasure to help the computer user away from the tracking on the Internet.

Are there any risks involved?
There is no any risk involved for participants.

Will my participation be confidential?
The name of the participants will not be taken and participation will be kept anonymous. All data will be safe in a protected computer. All will be destroyed one year after the research is finished.

What happens if I change my mind?
You have the right to withdraw from doing the questionnaire or interview at any time.

What happens if something goes wrong?
If you have any concern or complaint with this study, please contacts me (Sakchan Luangmaneerote: sl8e14@ soton.ac.uk).

Where can I get more information?
If you would like to get more information about this study, please feel free to contact me (Sakchan Luangmaneerote) anytime at sl8e14@soton.ac.uk

B.3 Debriefing Plan

Study title: A study of side-effects of using fingerprint countermeasures installed on a web browser

Investigator name: Sakchan Luangmaneerote
Study reference: 30576
Ethics reference: **ERGO/**FPSE**/30576**

This study was an investigation into negative side-effects of a web browser that use a fingerprint countermeasure for fingerprinting prevention. The fingerprinting technique is a technique used for tracking users on the Internet without storing any files on the user computer. Many users attempt to protect their privacy by installing fingerprint countermeasure on their web browser. The fingerprint countermeasures might protect the fingerprint tracking but there are side-effects of using. Side-effects are a result from inhibiting the operational mechanism of fingerprinting technique with different techniques.

So in this stage, we would like to observe side-effects of using each fingerprint countermeasure. The study selects Thai student, at Rajamangala University of Technology Isan in Thailand, who are skilled in the use of computer to observe side-effects while navigating. The participants would be invited to use a web browser installing a fingerprint countermeasure. The result of using would reflect the user satisfaction to the fingerprint countermeasure that is used. The participants need to answer questionnaires along with surfing a website. The survey will measure each question with a 5-point Likert scale, ranging from '1' (very displeased) to '5' (very pleased). There are two sections to the survey. The first will record the user's profile information such as age and gender. The second section will record the user's perceptions while using the fingerprint countermeasure. The user's satisfaction will be assessed as follows

- The quality of contents: Fonts and screen sizes should be displayed correctly, and users cannot observe any unusual website operation while navigating websites.
- The quality of functionality: All plugins should operate normally (such as pdf, video, audio)
- The quality of navigation: The user can navigate websites smoothly. There are no annoying delays introduced while surfing the Internet.
- The quality of Login: The login system should work normally and present no problems for users as they log into websites

During the conduct of survey, if the question induces any anxiety stress or other harmful psychological states on a momentary basis, please contact following places

Ruamphat Hospital Dr.ANANs
Address 312/1 Thetsaban 1 Rd, Tambon Nai Mueang, Amphoe Mueang Surin, Chang Wat Surin 32000, Thailand

Phone +66 44 513 638

We anticipate that participants will give valuable information which will be of use towards helping the Internet user away from being tracked on the Internet.

If you have any questions regarding this study, please contact Sakchan Luangmaneerote at the following e-mail address sl8e14@soton.ac.uk, Ed Zaluska (ejz@ecs.soton.ac.uk), Leslie Carr (lac@ecs.soton.ac.uk).

B.4 Permission for conducting survey

University of Southampton

27 June 2017

Dear Dean Assistant Professor Samnou Saowakoon

I would like to ask your permission to allow me to conduct a survey in your Faculty of Agriculture and Technology, Rajamangala University of Technology, Surin Campus, Thailand. This survey is a part of my PhD thesis "**Defense of Browser Fingerprinting Technique with FP-prevention**" I would conduct the survey in the department of computer and technology.

The survey would spend about 20 minutes and would arrange at a time convenient to students. Participation in the survey is entirely voluntary, and there are no anticipated risks to participation in this study. All information provided will be kept in confidentiality and would be used only for academic purposes.

If you agree, please kindly sign below acknowledging your consent and permission for me to conduct this study at your university.

Sincerely,

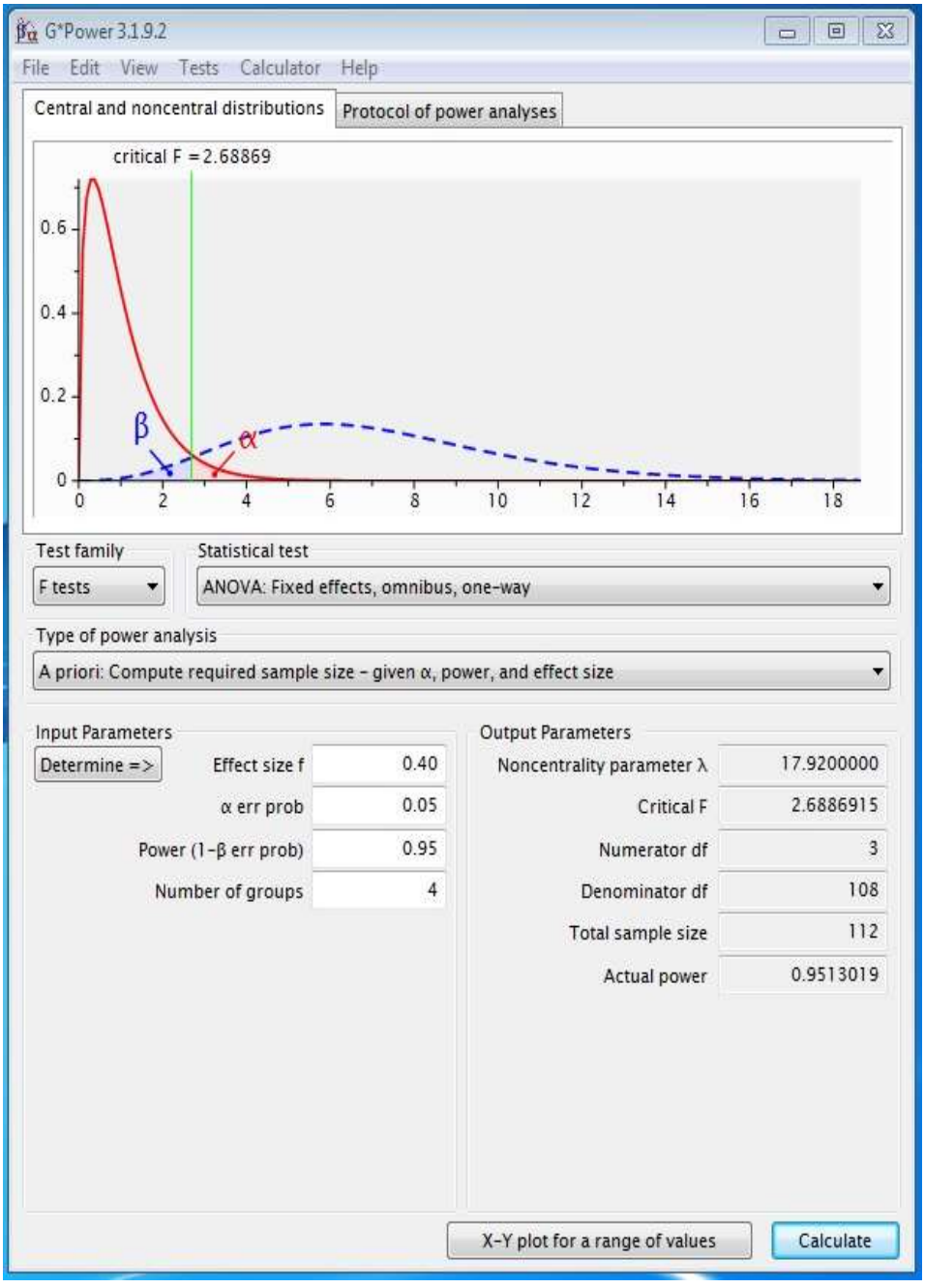
Sakchan Luangmaneerote

ECS PhD student

Approved by:

Asst. Prof. Dr. Samnou Saowakoon		29 ก.ย. 2560
Printed name and title	Signature	Date

B.5 Calculating an expected sample size for G*power



Appendix C Results of one-way ANOVA

Descriptives

functional experience

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean
					Lower Bound
FP-block	30	3.0444	.88293	.16120	2.7148
stop_fingerprinting	30	3.3667	.91224	.16655	3.0260
Fireglove	30	2.9778	.69857	.12754	2.7169
FP_prevention	30	4.4167	.63540	.11601	4.1794
Total	120	3.4514	.97216	.08875	3.2757

Descriptives

functional experience

	95% Confidence Interval for Mean	Minimum	Maximum
	Upper Bound		
FP-block	3.3741	1.50	4.67
stop_fingerprinting	3.7073	1.33	4.67
Fireglove	3.2386	2.00	4.33
FP_prevention	4.6539	3.00	5.00
Total	3.6271	1.33	5.00

Appendix C

ANOVA

functional experience

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	39.866	3	13.289	21.232	.000
Within Groups	72.601	116	.626		
Total	112.466	119			

Post Hoc Tests

Multiple Comparisons

Dependent Variable: functional experience

Tukey HSD

(I) countermeasure	(J) countermeasure	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval Lower Bound
FP-block	stop_fingerprinting	-.32222	.20427	.395	-.8547
	Fireglove	.06667	.20427	.988	-.4658
	FP_prevention	-1.37222*	.20427	.000	-1.9047
stop_fingerprinting	FP-block	.32222	.20427	.395	-.2102
	Fireglove	.38889	.20427	.232	-.1436
	FP_prevention	-1.05000*	.20427	.000	-1.5825
Fireglove	FP-block	-.06667	.20427	.988	-.5991
	stop_fingerprinting	-.38889	.20427	.232	-.9213
	FP_prevention	-1.43889*	.20427	.000	-1.9713
FP_prevention	FP-block	1.37222*	.20427	.000	.8398
	stop_fingerprinting	1.05000*	.20427	.000	.5175

	Fireglove	1.43889*	.20427	.000	.9064

Multiple Comparisons

Dependent Variable: functional experience

Tukey HSD

(I) countermeasure	(J) countermeasure	95% Confidence Interval Upper Bound
FP-block	stop_fingerprinting	.2102
	Fireglove	.5991
	FP_prevention	-.8398
stop_fingerprinting	FP-block	.8547
	Fireglove	.9213
	FP_prevention	-.5175
Fireglove	FP-block	.4658
	stop_fingerprinting	.1436
	FP_prevention	-.9064
FP_prevention	FP-block	1.9047
	stop_fingerprinting	1.5825
	Fireglove	1.9713

*. The mean difference is significant at the 0.05 level.

Appendix C

Homogeneous Subsets

functional experience

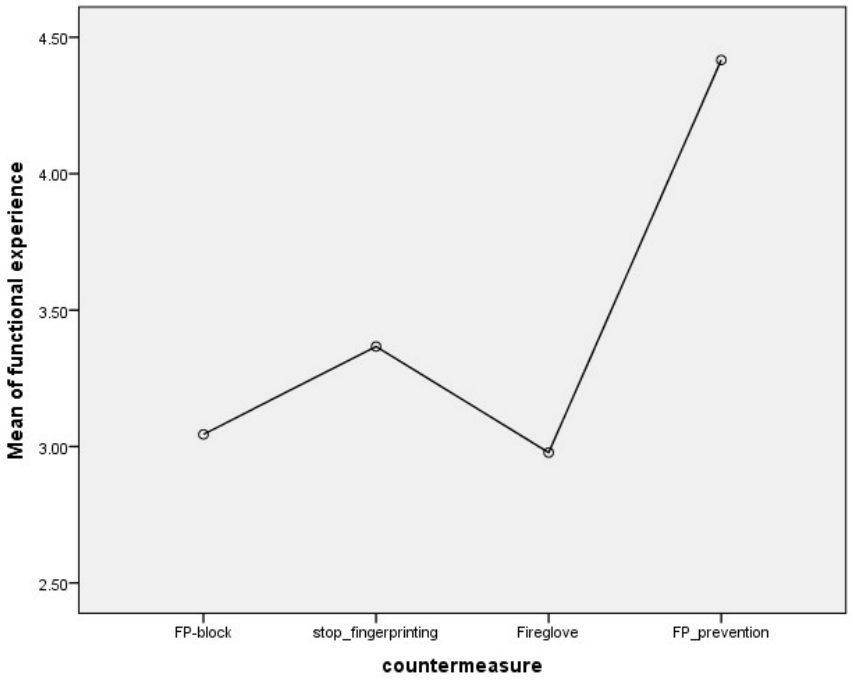
Tukey HSD[a]

countermeasure	N	Subset for alpha = 0.05	
		1	2
Fireglove	30	2.9778	
FP-block	30	3.0444	
stop_fingerprinting	30	3.3667	
FP_prevention	30		4.4167
Sig.		.232	1.000

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 30.000.

Means Plots



Descriptives

Dominance

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean
					Lower Bound
FP-block	30	2.8500	1.04345	.19051	2.4604
stop_fingerprinting	30	3.3833	1.30439	.23815	2.8963
Fireglove	30	2.5667	.98902	.18057	2.1974
FP_prevention	30	4.5000	.58722	.10721	4.2807
Total	120	3.3250	1.24625	.11377	3.0997

Descriptives

Dominance

	95% Confidence Interval for Mean	Minimum	Maximum
	Upper Bound		
FP-block	3.2396	1.00	4.50
stop_fingerprinting	3.8704	1.00	5.00
Fireglove	2.9360	1.00	4.50
FP_prevention	4.7193	3.00	5.00
Total	3.5503	1.00	5.00

ANOVA

Dominance

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	65.542	3	21.847	21.246	.000

Within Groups	119.283	116	1.028		
Total	184.825	119			

Post Hoc Tests

Multiple Comparisons

Dependent Variable: Dominance

Tukey HSD

(I) countermeasure	(J) countermeasure	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval Lower Bound
FP-block	stop_fingerprinting	-.53333	.26183	.181	-1.2158
	Fireglove	.28333	.26183	.701	-.3992
	FP_prevention	-1.65000*	.26183	.000	-2.3325
stop_fingerprinting	FP-block	.53333	.26183	.181	-.1492
	Fireglove	.81667*	.26183	.012	.1342
	FP_prevention	-1.11667*	.26183	.000	-1.7992
Fireglove	FP-block	-.28333	.26183	.701	-.9658
	stop_fingerprinting	-.81667*	.26183	.012	-1.4992
	FP_prevention	-1.93333*	.26183	.000	-2.6158
FP_prevention	FP-block	1.65000*	.26183	.000	.9675
	stop_fingerprinting	1.11667*	.26183	.000	.4342
	Fireglove	1.93333*	.26183	.000	1.2508

Multiple Comparisons

Dependent Variable: Dominance

Tukey HSD

(I) countermeasure	(J) countermeasure	95% Confidence Interval Upper Bound
FP-block	stop_fingerprinting	.1492
	Fireglove	.9658
	FP_prevention	-.9675
stop_fingerprinting	FP-block	1.2158
	Fireglove	1.4992
	FP_prevention	-.4342
Fireglove	FP-block	.3992
	stop_fingerprinting	-.1342
	FP_prevention	-1.2508
FP_prevention	FP-block	2.3325
	stop_fingerprinting	1.7992
	Fireglove	2.6158

*. The mean difference is significant at the 0.05 level.

Homogeneous Subsets

Dominance

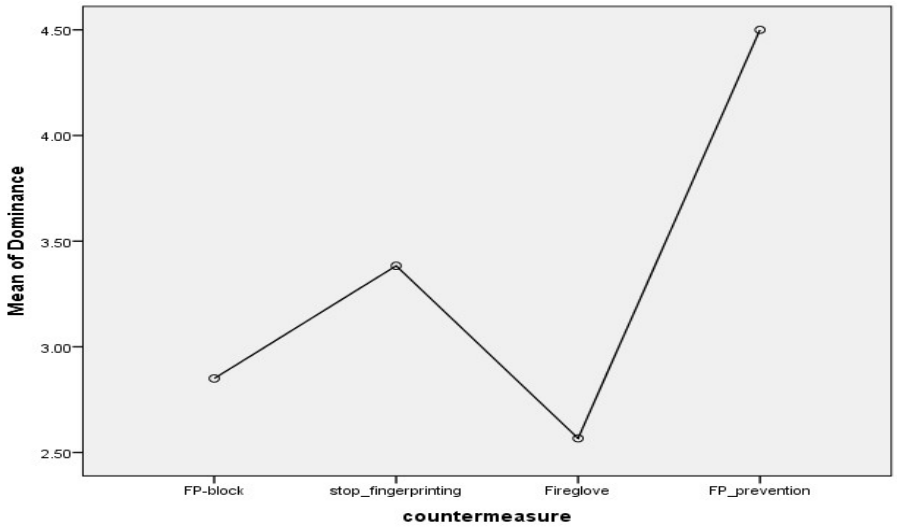
Tukey HSD[a]

countermeasure	N	Subset for alpha = 0.05		
		1	2	3
Fireglove	30	2.5667		
FP-block	30	2.8500	2.8500	
stop_fingerprinting	30		3.3833	
FP_prevention	30			4.5000
Sig.		.701	.181	1.000

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 30.000.

Means Plots



Descriptives

Social experience

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean
					Lower Bound
FP-block	30	2.4000	.97733	.17844	2.0351
stop_fingerprinting	30	3.0167	1.05441	.19251	2.6229
Fireglove	30	1.9333	.94443	.17243	1.5807
FP_prevention	30	4.5000	.50855	.09285	4.3101
Total	120	2.9625	1.31437	.11999	2.7249

Descriptives

Social experience

	95% Confidence Interval for Mean		
	Upper Bound	Minimum	Maximum
FP-block	2.7649	1.00	4.50
stop_fingerprinting	3.4104	1.50	4.50
Fireglove	2.2860	1.00	4.50
FP_prevention	4.6899	3.00	5.00
Total	3.2001	1.00	5.00

Appendix C

ANOVA

Social experience

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	112.273	3	37.424	46.526	.000
Within Groups	93.308	116	.804		
Total	205.581	119			

Post Hoc Tests

Multiple Comparisons

Dependent Variable: Social experience

Tukey HSD

(I) countermeasure	(J) countermeasure	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval Lower Bound
FP-block	stop_fingerprinting	-.61667*	.23157	.043	-1.2203
	Fireglove	.46667	.23157	.188	-.1370
	FP_prevention	-2.10000*	.23157	.000	-2.7036
stop_fingerprinting	FP-block	.61667*	.23157	.043	.0130
	Fireglove	1.08333*	.23157	.000	.4797
	FP_prevention	-1.48333*	.23157	.000	-2.0870
Fireglove	FP-block	-.46667	.23157	.188	-1.0703
	stop_fingerprinting	-1.08333*	.23157	.000	-1.6870
	FP_prevention	-2.56667*	.23157	.000	-3.1703
FP_prevention	FP-block	2.10000*	.23157	.000	1.4964
	stop_fingerprinting	1.48333*	.23157	.000	.8797
	Fireglove	2.56667*	.23157	.000	1.9630

Multiple Comparisons

Dependent Variable: Social experience

Tukey HSD

(I) countermeasure	(J) countermeasure	95% Confidence Interval Upper Bound
FP-block	stop_fingerprinting	-.0130
	Fireglove	1.0703
	FP_prevention	-1.4964
stop_fingerprinting	FP-block	1.2203
	Fireglove	1.6870
	FP_prevention	-.8797
Fireglove	FP-block	.1370
	stop_fingerprinting	-.4797
	FP_prevention	-1.9630
FP_prevention	FP-block	2.7036
	stop_fingerprinting	2.0870
	Fireglove	3.1703

*. The mean difference is significant at the 0.05 level.

Homogeneous Subsets

Social experience

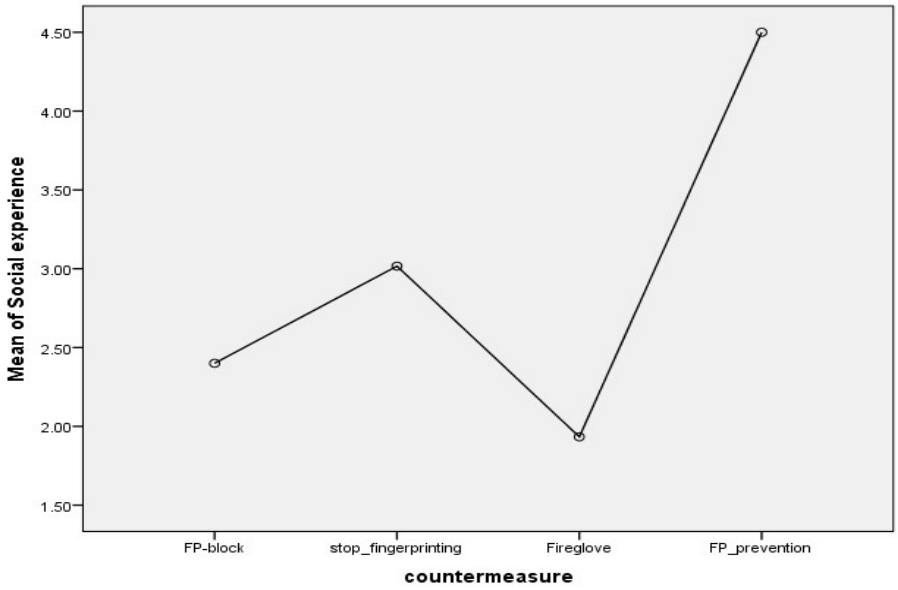
Tukey HSD[a]

countermeasure	N	Subset for alpha = 0.05		
		1	2	3
Fireglove	30	1.9333		
FP-block	30	2.4000		
stop_fingerprinting	30		3.0167	
FP_prevention	30			4.5000
Sig.		.188	1.000	1.000

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 30.000.

Means Plots



Descriptives

Hedonic experience

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean
					Lower Bound
FP-block	30	2.3833	.94398	.17235	2.0308
stop_fingerprinting	30	2.7500	1.04840	.19141	2.3585
Fireglove	30	2.1500	.88230	.16108	1.8205
FP_prevention	30	4.4333	.66609	.12161	4.1846
Total	120	2.9292	1.26075	.11509	2.7013

Descriptives

Hedonic experience

	95% Confidence Interval for Mean	Minimum	Maximum
	Upper Bound		
FP-block	2.7358	1.00	4.50
stop_fingerprinting	3.1415	1.00	4.50
Fireglove	2.4795	1.00	4.00
FP_prevention	4.6821	3.00	5.00
Total	3.1571	1.00	5.00

ANOVA

Hedonic experience

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	95.990	3	31.997	39.842	.000
Within Groups	93.158	116	.803		
Total	189.148	119			

Post Hoc Tests

Multiple Comparisons

Dependent Variable: Hedonic experience

Tukey HSD

(I) countermeasure	(J) countermeasure	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval
					Lower Bound
FP-block	stop_fingerprinting	-.36667	.23139	.391	-.9698
	Fireglove	.23333	.23139	.745	-.3698
	FP_prevention	-2.05000*	.23139	.000	-2.6531
stop_fingerprinting	FP-block	.36667	.23139	.391	-.2365
	Fireglove	.60000	.23139	.052	-.0031
	FP_prevention	-1.68333*	.23139	.000	-2.2865
Fireglove	FP-block	-.23333	.23139	.745	-.8365
	stop_fingerprinting	-.60000	.23139	.052	-1.2031
	FP_prevention	-2.28333*	.23139	.000	-2.8865
FP_prevention	FP-block	2.05000*	.23139	.000	1.4469
	stop_fingerprinting	1.68333*	.23139	.000	1.0802

	Fireglove	2.28333*	.23139	.000	1.6802

Multiple Comparisons

Dependent Variable: Hedonic experience

Tukey HSD

(I) countermeasure	(J) countermeasure	95% Confidence Interval Upper Bound
FP-block	stop_fingerprinting	.2365
	Fireglove	.8365
	FP_prevention	-1.4469
stop_fingerprinting	FP-block	.9698
	Fireglove	1.2031
	FP_prevention	-1.0802
Fireglove	FP-block	.3698
	stop_fingerprinting	.0031
	FP_prevention	-1.6802
FP_prevention	FP-block	2.6531
	stop_fingerprinting	2.2865
	Fireglove	2.8865

*. The mean difference is significant at the 0.05 level.

Homogeneous Subsets

Hedonic experience

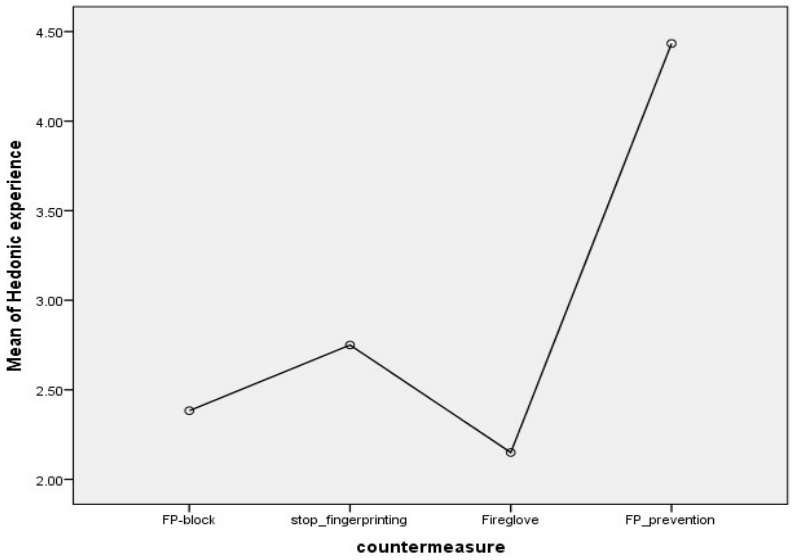
Tukey HSD[a]

countermeasure	N	Subset for alpha = 0.05	
		1	2
Fireglove	30	2.1500	
FP-block	30	2.3833	
stop_fingerprinting	30	2.7500	
FP_prevention	30		4.4333
Sig.		.052	1.000

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 30.000.

Means Plots



Descriptives

Pleasure

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean
					Lower Bound
FP-block	30	2.1000	.88474	.16153	1.7696
stop_fingerprinting	30	3.1167	1.05604	.19281	2.7223
Fireglove	30	2.1167	.75067	.13705	1.8364
FP_prevention	30	4.4333	.63968	.11679	4.1945
Total	120	2.9417	1.27217	.11613	2.7117

Descriptives

Pleasure

	95% Confidence Interval for Mean	Minimum	Maximum
	Upper Bound		
FP-block	2.4304	1.00	4.00
stop_fingerprinting	3.5110	1.50	4.50
Fireglove	2.3970	1.00	4.00
FP_prevention	4.6722	3.00	5.00
Total	3.1716	1.00	5.00

Appendix C

ANOVA

Pleasure

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	109.342	3	36.447	50.785	.000
Within Groups	83.250	116	.718		
Total	192.592	119			

Post Hoc Tests

Multiple Comparisons

Dependent Variable: Pleasure

Tukey HSD

(I) countermeasure	(J) countermeasure	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval
					Lower Bound
FP-block	stop_fingerprinting	-1.01667*	.21873	.000	-1.5868
	Fireglove	-.01667	.21873	1.000	-.5868
	FP_prevention	-2.33333*	.21873	.000	-2.9035
stop_fingerprinting	FP-block	1.01667*	.21873	.000	.4465
	Fireglove	1.00000*	.21873	.000	.4298
	FP_prevention	-1.31667*	.21873	.000	-1.8868
Fireglove	FP-block	.01667	.21873	1.000	-.5535
	stop_fingerprinting	-1.00000*	.21873	.000	-1.5702
	FP_prevention	-2.31667*	.21873	.000	-2.8868
FP_prevention	FP-block	2.33333*	.21873	.000	1.7632
	stop_fingerprinting	1.31667*	.21873	.000	.7465
	Fireglove	2.31667*	.21873	.000	1.7465

Multiple Comparisons

Dependent Variable: Pleasure

Tukey HSD

(I) countermeasure	(J) countermeasure	95% Confidence Interval Upper Bound
FP-block	stop_fingerprinting	-.4465
	Fireglove	.5535
	FP_prevention	-1.7632
stop_fingerprinting	FP-block	1.5868
	Fireglove	1.5702
	FP_prevention	-.7465
Fireglove	FP-block	.5868
	stop_fingerprinting	-.4298
	FP_prevention	-1.7465
FP_prevention	FP-block	2.9035
	stop_fingerprinting	1.8868
	Fireglove	2.8868

*. The mean difference is significant at the 0.05 level.

Homogeneous Subsets

Pleasure

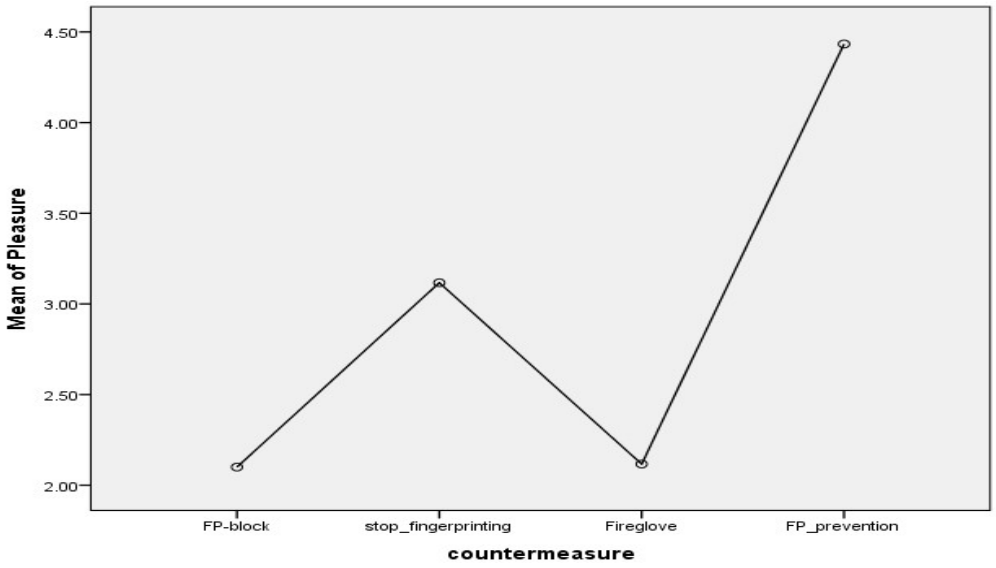
Tukey HSD[a]

countermeasure	N	Subset for alpha = 0.05		
		1	2	3
FP-block	30	2.1000		
Fireglove	30	2.1167		
stop_fingerprinting	30		3.1167	
FP_prevention	30			4.4333
Sig.		1.000	1.000	1.000

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 30.000.

Means Plots



Descriptives

Word of Mouth

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean
					Lower Bound
FP-block	30	2.3667	1.06620	.19466	1.9685
stop_fingerprinting	30	2.7167	1.09610	.20012	2.3074
Fireglove	30	2.1333	1.09019	.19904	1.7263
FP_prevention	30	4.4833	.60861	.11112	4.2561
Total	120	2.9250	1.34516	.12280	2.6819

Descriptives

Word of Mouth

	95% Confidence Interval for Mean	Minimum	Maximum
	Upper Bound		
FP-block	2.7648	1.00	5.00
stop_fingerprinting	3.1260	1.00	4.50
Fireglove	2.5404	1.00	5.00
FP_prevention	4.7106	3.00	5.00
Total	3.1681	1.00	5.00

Appendix C

ANOVA

Word of Mouth

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	102.308	3	34.103	35.003	.000
Within Groups	113.017	116	.974		
Total	215.325	119			

Post Hoc Tests

Multiple Comparisons

Dependent Variable: Word of Mouth

Tukey HSD

(I) countermeasure	(J) countermeasure	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval
					Lower Bound
FP-block	stop_fingerprinting	-.35000	.25486	.519	-1.0143
	Fireglove	.23333	.25486	.797	-.4310
	FP_prevention	-2.11667*	.25486	.000	-2.7810
stop_fingerprinting	FP-block	.35000	.25486	.519	-.3143
	Fireglove	.58333	.25486	.107	-.0810
	FP_prevention	-1.76667*	.25486	.000	-2.4310
Fireglove	FP-block	-.23333	.25486	.797	-.8977
	stop_fingerprinting	-.58333	.25486	.107	-1.2477
	FP_prevention	-2.35000*	.25486	.000	-3.0143
FP_prevention	FP-block	2.11667*	.25486	.000	1.4523
	stop_fingerprinting	1.76667*	.25486	.000	1.1023
	Fireglove	2.35000*	.25486	.000	1.6857

Multiple Comparisons

Dependent Variable: Word of Mouth

Tukey HSD

(I) countermeasure	(J) countermeasure	95% Confidence Interval Upper Bound
FP-block	stop_fingerprinting	.3143
	Fireglove	.8977
	FP_prevention	-1.4523
stop_fingerprinting	FP-block	1.0143
	Fireglove	1.2477
	FP_prevention	-1.1023
Fireglove	FP-block	.4310
	stop_fingerprinting	.0810
	FP_prevention	-1.6857
FP_prevention	FP-block	2.7810
	stop_fingerprinting	2.4310
	Fireglove	3.0143

*. The mean difference is significant at the 0.05 level.

Homogeneous Subsets

Word of Mouth

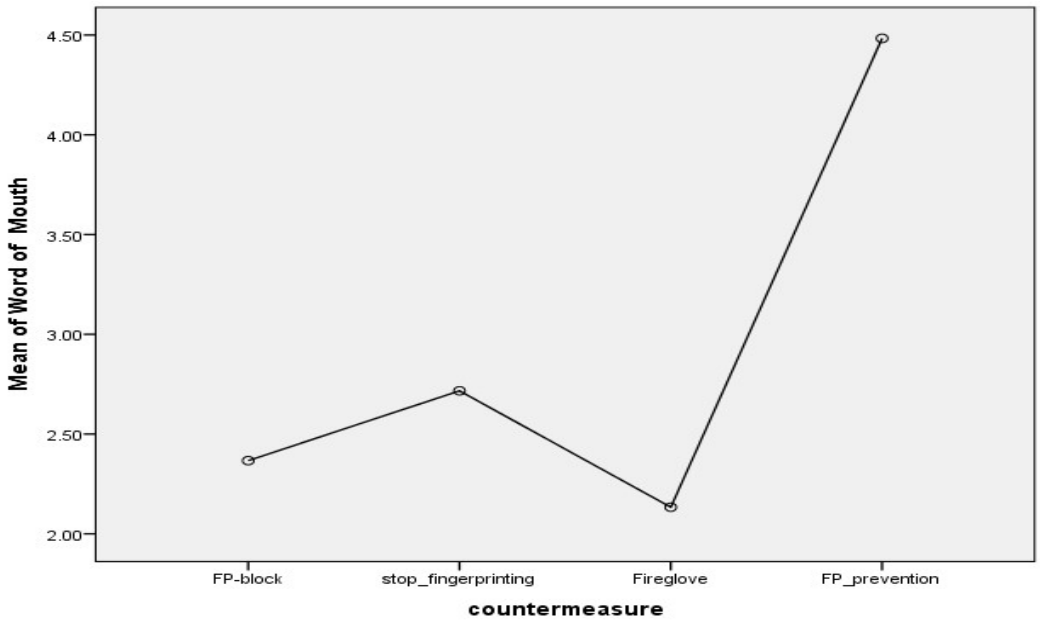
Tukey HSD[a]

countermeasure	N	Subset for alpha = 0.05	
		1	2
Fireglove	30	2.1333	
FP-block	30	2.3667	
stop_fingerprinting	30	2.7167	
FP_prevention	30		4.4833
Sig.		.107	1.000

Means for groups in homogeneous subsets are displayed.

a. Uses Harmonic Mean Sample Size = 30.000.

Means Plots



Appendix D Result of t-test

D.1 Result of JSBench

Group Statistics

	status	N	Mean	Std. Deviation	Std. Error Mean
mean	unmodified	30	75.5023	1.63959	.29935
	modified	30	73.6513	1.58359	.28912

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means	
		F	Sig.	t	df
mean	Equal variances assumed	.100	.753	4.448	58
	Equal variances not assumed			4.448	57.930

Independent Samples Test

		t-test for Equality of Means			
		Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference
					Lower
mean	Equal variances assumed	.000	1.85100	.41617	1.01794
	Equal variances not assumed	.000	1.85100	.41617	1.01792

Independent Samples Test

		t-test for Equality of Means
		95% Confidence Interval of the Difference
		Upper
mean	Equal variances assumed	2.68406
	Equal variances not assumed	2.68408

D.2 Result of Kraken

Group Statistics

	type_benchmark	N	Mean	Std. Deviation	Std. Error Mean
mean	unmodified	30	1159.7167	4.06449	.74207
	modified	30	1161.6600	3.32634	.60730

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means
		F	Sig.	t
mean	Equal variances assumed	.474	.494	-2.027
	Equal variances not assumed			-2.027

Independent Samples Test

		t-test for Equality of Means		
		df	Sig. (2-tailed)	Mean Difference
mean	Equal variances assumed	58	.047	-1.94333
	Equal variances not assumed	55.817	.047	-1.94333

Independent Samples Test

		t-test for Equality of Means		
			95% Confidence Interval of the Difference	
		Std. Error Difference	Lower	Upper
mean	Equal variances assumed	.95890	-3.86278	-.02389
	Equal variances not assumed	.95890	-3.86438	-.02229

D.3 Result of Sunspider

Group Statistics

	FP_prevention	N	Mean	Std. Deviation	Std. Error Mean
mean	unmodified	30	237.5300	4.90476	.89548
	modified	30	237.4000	7.87804	1.43833

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means	
		F	Sig.	t	df
mean	Equal variances assumed	1.306	.258	.077	58
	Equal variances not assumed			.077	48.545

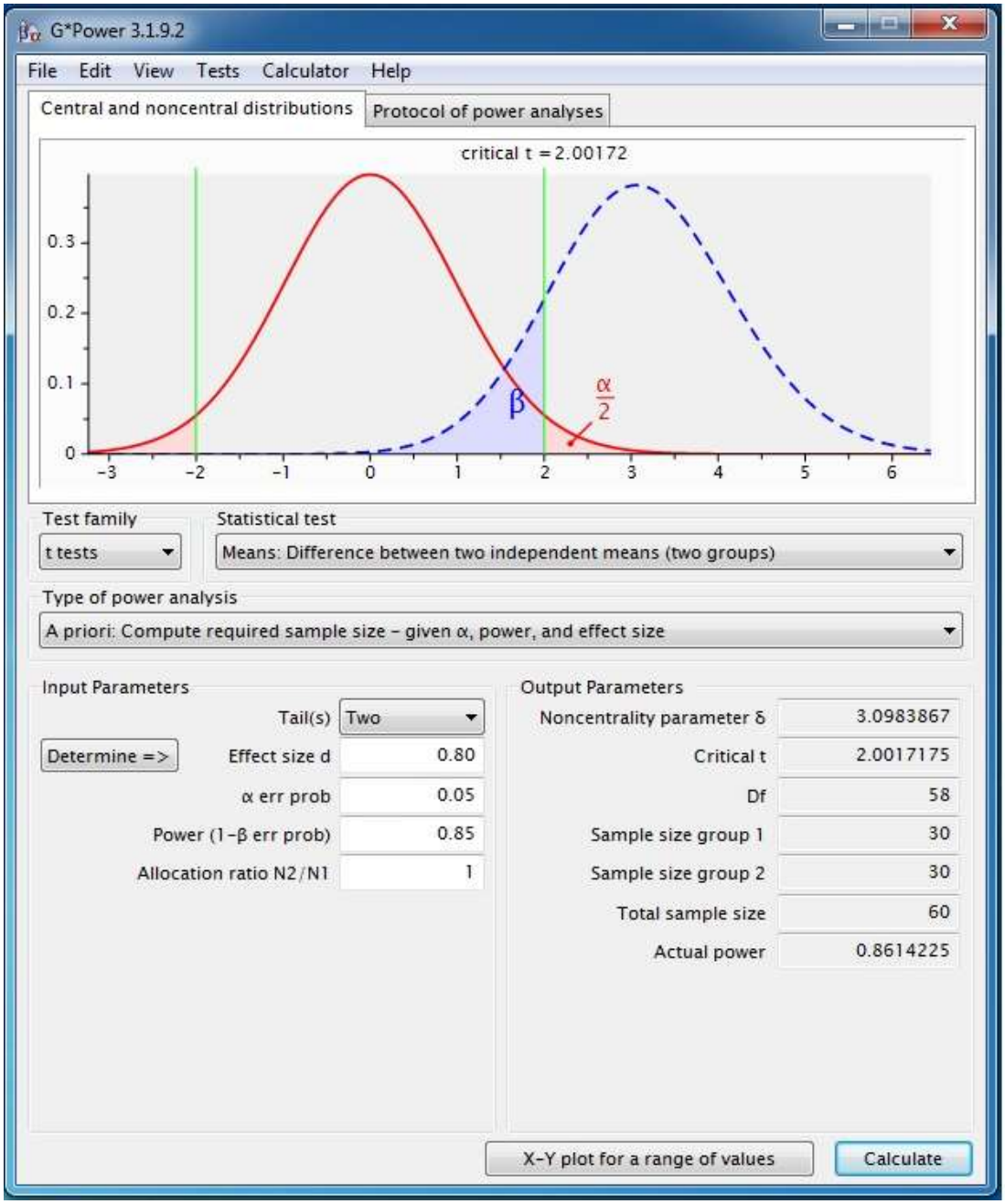
Independent Samples Test

		t-test for Equality of Means		
		Sig. (2-tailed)	Mean Difference	Std. Error Difference
mean	Equal variances assumed	.939	.13000	1.69431
	Equal variances not assumed	.939	.13000	1.69431

Independent Samples Test

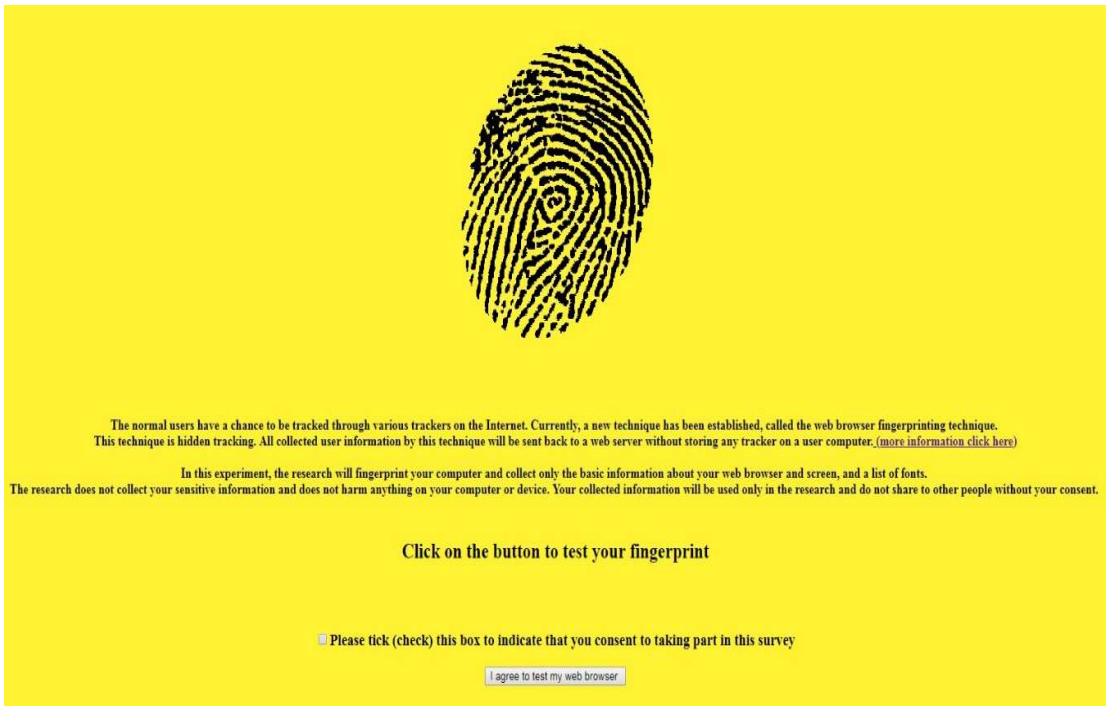
		t-test for Equality of Means	
		95% Confidence Interval of the Difference	
		Lower	Upper
mean	Equal variances assumed	-3.26152	3.52152
	Equal variances not assumed	-3.27564	3.53564

D.4 Using G*power to calculate

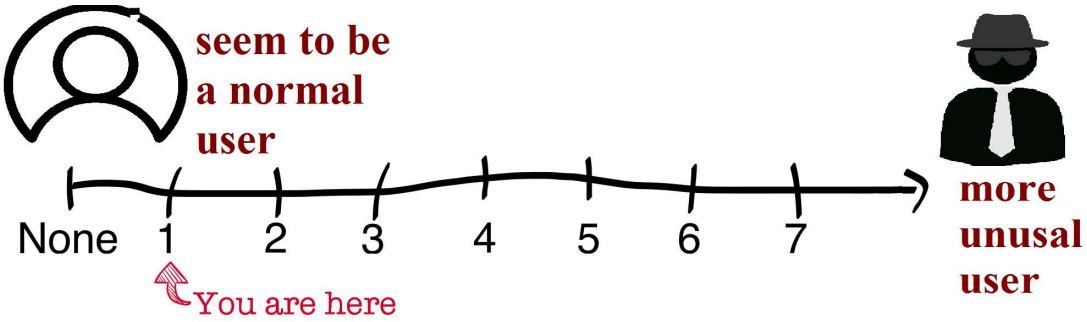


Appendix E Fingerprinting survey

E.1 The main web page



E.2 The web page shows result of fingerprinting along with analysis of combination of user information



E.3 Debrief plan

Debriefing Plan

Study title: A study about the basic technique of the browser fingerprinting technique.

Investigator name: Sakchan Luangmaneerote

Study reference: 41395

Ethics reference: **ERGO/**FPSE**/**41395

This study was an investigation into the workflow of basic browser fingerprinting technique. The browser fingerprinting technique is a technique used for tracking users on the Internet without storing any files on the user computer.

So in this stage, we would like to observe user's data used through the basic browser fingerprinting technique on the Internet. The study are required to collect the real basic information of a user computer, no collecting a personal information. The participants would be invited to visit a website (http:// http://www.pleasefingerprintme2.org/index.php) installing a browser fingerprinting technique code. The result of visiting website is user's data. The collected user's data will be installed in the server's database for analysis later. The participants need to read a background information of the browser fingerprinting technique first before ticking checkbox for consent at a main web page. After participants tick checkbox for consent, they need to click button for allowing code of the browser fingerprinting technique to proceed.

After clicking button, the operation of browser fingerprinting technique will take time within few second to collect the user's data. The collected basic information storing in database will be shown the user's screen.

We anticipate that participants will give valuable information which will be of use towards helping the Internet user away from being tracked on the Internet.

If you have any questions regarding this study, please contact Sakchan Luangmaneerote at the following e-mail address sl8e14@soton.ac.uk, Ed Zaluska (ejz@ecs.soton.ac.uk), Leslie Carr (lac@ecs.soton.ac.uk).

E.4 Risk assessment

October 04, 2017

Risk Assessment Form

- Please see Guidance Notes for completing the risk assessment form at the end of this document.

Researcher's name: Sakchan Luangmaneerote

Part 1 – Dissertation/project activities
What do you intend to do? The research is attempting to collect a basic information of user computer through the browser fingerprinting technique. It is essential to know how the basic fingerprinting technique work in practice. In order to collect the data, the participant will be invited to visit the fingerprinting web site (http://www.pleasefingerprintme2.org/index.php). The participants are required to read the basic information of the browser fingerprinting technique before ticking box for consent. After a participant clicks a button for fingerprint, the basic information of user computer (e.g., name of browser, version of browser, language) will be stored on the database automatically. The survey did not store any personal information of a participant. All information collected is only a basic information of user computer such as operating system, list of fonts or list of plugins.

Will this involve collection of information from other people? 1. A computer user 2. Researcher
If relevant, what location/s is/are involved? 1. Department of computer technology, Faculty of agriculture and technology, Rajamangala University of Technology Isan (Thailand), Surin campus. 2. University of Southampton University Road Southampton SO17 1BJ
Will you be working alone or with others? Alone
Part 2 – Potential safety issues / risk assessment.
Potential safety issues arising from proposed activity? No safety issues arising during questionnaire
Person/s likely to be affected?

No any affected to participants
Likelihood of risk? 0%
Part 3 – Precautions / risk reduction
Existing precautions: Precaution is no need
Proposed risk reduction strategies if existing precautions are not adequate: Risk reduction strategy is no need

Part 4 – International Travel

If you intend to travel overseas to carry out fieldwork then you must carry out a risk assessment for each trip you make and attach a copy of the International Travel form to this document

Download the Risk Assessment for International Travel Form

Guidelines on risk assessment for international travel at can be located at:

www.southampton.ac.uk/socscinet/safety ("risk assessment" section).

Before undertaking international travel and overseas visits all students must:

- Ensure a risk assessment has been undertaken for all journeys including to conferences and visits to other Universities and organisations. This is University policy and is not optional.
- Consult the University Finance/Insurance website for information on travel and insurance. Ensure that you take a copy of the University travel insurance information with you and know what to do if you should need medical assistance.
- Obtain from Occupational Health Service advice on any medical requirements for travel to areas to be visited.
- Ensure next of kin are aware of itinerary, contact person and telephone number at the University.
- Where possible arrange to be met by your host on arrival.

If you are unsure if you are covered by the University insurance scheme for the trip you are undertaking and for the country/countries you intend visiting, then you should contact the University's Insurance Office at insure@soton.ac.uk and check the Foreign and Commonwealth Office website.

Risk Assessment Form for International Travel attached	**NO**

E.5 Data Protection Plan

DATA PROTECTION PLAN

Study title: Survey information about a web browser

Investigator name: Sakchan Luangmaneerote

Study reference: 41395

Ethics reference: ERGO/FPSE/41395

Collected Data through web: All data confidentially kept in a password-protected computer at the University of Southampton. Investigator will be responsible for a secure storage and retention of the datasets during the period is a maximum of 1 years after the completion of this study, following the recommendations set out in the University of Southampton Research Data Management Policy. All of files and transcription files will be destroyed 1 year after the study is finished.

The data will be processed in accordance with the rights of the participants because they will have the right to access, correct, and/or withdraw their data at any time and for any reason. Participants will be able to exercise their rights by contacting the investigator (e-mail: sl8e14@soton.ac.uk) or the project supervisor (e-mail: Ed Zaluska, ejz@ecs.soton.ac.uk and Professor Leslie Carr, lac@ecs.soton.ac.uk).

At all stages the data will be anonymous and the name of participants will not be used at any times.

Appendix F Summarizing results of visiting fingerprinters

Visiting FingerprintJS2

countermeasures	100		200		300		400		500	
	duplicate fingerprinting ID	unique fingerprinting ID	duplicate fingerprinting ID	unique fingerprinting ID	duplicate fingerprinting ID	unique fingerprinting ID	duplicate fingerprinting ID	unique fingerprinting ID	duplicate fingerprinting ID	unique fingerprinting ID
FP-prevention	44	55	74	126	116	184	146	254	189	311
Stop-fingerprinting	2	98	3	197	4	296	9	391	11	489
Fireglove	100	0	200	0	300	0	400	0	500	0
FP-Block	0	100	0	200	0	300	0	400	0	500

Visiting fingerprint.pet-portal.eu

countermeasures	100		200		300		400		500	
	duplicate fingerprinting ID	unique fingerprinting ID	duplicate fingerprinting ID	unique fingerprinting ID	duplicate fingerprinting ID	unique fingerprinting ID	duplicate fingerprinting ID	unique fingerprinting ID	duplicate fingerprinting ID	unique fingerprinting ID
FP-prevention	10	90	43	157	91	209	146	254	206	294
Stop-fingerprinting	0	100	0	200	0	300	1	399	2	498
Fireglove	100	0	200	0	300	0	400	0	500	0
FP-Block	16	84	44	156	93	207	158	242	230	270

Visiting hidester.com

countermeasures	100		200		300		400		500	
	duplicate fingerprinting ID	unique fingerprinting ID	duplicate fingerprinting ID	unique fingerprinting ID	duplicate fingerprinting ID	unique fingerprinting ID	duplicate fingerprinting ID	unique fingerprinting ID	duplicate fingerprinting ID	unique fingerprinting ID
FP-prevention	36	64	64	136	104	196	149	251	184	316
Stop-fingerprinting	26	74	55	145	95	205	137	263	185	315
Fireglove	100	0	200	0	300	0	400	0	500	0
FP-Block	1	99	1	199	1	299	1	399	1	499

Appendix H Smart fingerprinter

Usually, the browser fingerprinting technique requires a unique fingerprint ID to refer to the individual user. It is not surprising why many published research papers focus on the uniqueness of fingerprint ID (Eckersley, 2010, Boda et al., 2014). However, uniqueness of fingerprint ID is not sufficient in the long run because the browser's attributes can be changed by nature over time or with different fingerprint countermeasures. The browser's attributes can be changed for many reasons as follows.

1. **Automatic change**

   These attributes change automatically without user awareness such as upgrading software, updating a web browser or plugins.

2. **Context-dependent change**

   Some attributes, such as Timezone, indirectly impact the browser's attributes, resulting from the user traveling to different locations.

3. **User-triggered change**

   Some attributes are changed by the user according to their preferences such as Do Not Track, cookie enabled, screen resolution.

4. **Changed by the countermeasure**

   Some browser's attributes are altered by the fingerprint countermeasure with the purpose of breaking the linkability of fingerprint such as User-agent, language or browser version.

When the fingerprinter collects fingerprints, it is feasible that obtained fingerprints derive from a prior visitor or a new visitor. The smart fingerprinter can link fingerprints by comparing the collected fingerprint ID with relevant information stored in their database. The result could assist a smart fingerprinter to identify a user in case that a user's attributes are changed.

With the understanding about the nature of browser's attributes, a smart fingerprinter attempts to defeat a change of browser's attributes by training an algorithm or applying rules in their algorithms in order to project a probability of changed attributes in case of change by nature of browser attribute or by fingerprint countermeasures. It can be classified as follows.

1. Using rule-based linking algorithm

This approach sets rules to project a change of browser's attributes. This approach is based on a concept that some attributes are stable over time such as operating system or browser family, some are changed frequently such as browser version or User-agent. Understanding the frequency of change of browser attributes could establish rules to project a probability of changed attributes. For example, FP-STALKER proposed by Vastel et al. (2018) had defined rules as follows.

1) The operating system and browser family must be matching.
2) Browser version can be constant or upgraded, not downgraded.
3) Local storage, DNT and cookies should be constant.
4) Fonts can be changed such as adding fonts or deleting fonts.
5) The set of attributes is allowed to change with similarity ratio > 0.75 by using difflib.SequenceMatcher().ratio in function of Python library.
6) Resolution, timezone and encoding can be changed but allowing only one attribute to be changed at the same time among the three attributes.
7) The total number of attributes changing in rules 5 and 6 is required less than two.

The order of established rules by the rule-based linking algorithm is extremely crucial to observe the evaluation of fingerprint. The first rule of the algorithm will discard all candidates from a list in order to reduce the number of a comparison. In order to link an unknown fingerprint (f_u) to known fingerprint (f_k), this algorithm applies rules according to a known fingerprint taken from the set of known browser fingerprints (F). When the algorithm obtains the fingerprint instance, the fingerprint instance will be inspected by the established rules. If the fingerprint instance is not matching the established rules, the fingerprint instance will be thrown it away. In another case, if the fingerprint instance matches the established rules, the fingerprint instance will be added to a list as a potential candidate. In case that an unknown fingerprint (f_u) and known fingerprint (f_k) are exact, the algorithm will add the unknown fingerprint to a list as a matching candidate, *exact*. When the process of verifying rules is accomplished, the algorithm considers two lists of candidates. If *exact* is not empty, the algorithm examines if there is only one candidate or

if all candidates derive from the same browser instance. If the latter is the case, the algorithm will link an unknown fingerprint (f_u) with this fingerprint instance. Otherwise, the algorithm will add the new fingerprint as an unknown fingerprint (f_u). The rule-base algorithm can be summarized as below.

Rule-based Matching Algorithm
function FINGERPRINTMATCHING(*F, f_u*)
rules = {$rule_1$, …, $rule_6$}
candidates ← ∅
exact ← ∅
for $f_k \in F$ **do**
if *VERIFYRULES* (f_k , f_u , *rules*) **then**
if *nbDiff* = 0 **then**
exact ← *exact* ∪ ⟨ f_k ⟩
else
candidates ← *candidates* ∪ ⟨ f_k ⟩
end if
end if
end for
if | *exact* | > 0 **and** SAMEIDS (*exact*) **then**
return *exact* [0] *.id*
else if | *candidates* | > 0 **and** SAMEIDS (*candidates*) **then**
return *candidates*[0] *.id*
else
return GENERATENEWID()
else if
end function
function of SAMEIDS considers a list of candidates. This function will return true if all candidates share the said id, else returning false.

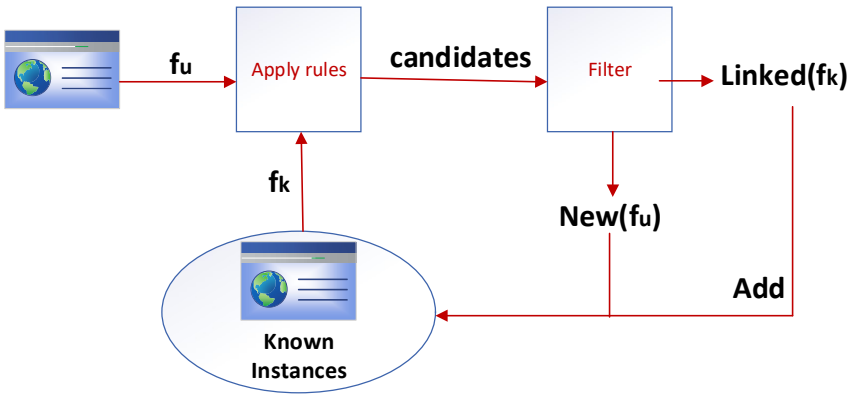


Figure H-1 rule-based linking algorithm

2. Using a hybrid linking algorithm

For this approach, the smart fingerprinter attempts to develop a new approach by combining the rule-based algorithm with machine learning to project a change of browser's attributes. With the difficulty of making a sophisticated rule to project a change of browser's attributes, this approach uses machine learning to assist prediction of changed attributes by combining with some rules of the rule-based algorithm into the hybrid linking algorithm in order to fix a weakness of the rule-base algorithm.

The initial step of this algorithm still uses some rules of rule-based algorithm. For the part of the machine learning, computing probability of unknown fingerprint (f_u) and known fingerprint (f_k) could be modeled as a binary classification. With this approach, the same browser instance and different browser instance will be predicted by two classes, applying a random forest algorithm to solve the binary classification. Each decision tree will make a prediction and votes in case that two browsers fingerprints derive from the same browser instance. The majority of votes will be chosen as a result.

Considering two fingerprints, $f_u \notin F$ and $f_k \in F$ in Figure H-2, the two set will be reduced into a single feature vector of M features $X = \langle x_1, x_2, \ldots, x_m \rangle$ while the random forest model will calculate the probability $P(f_u.id = f_k.id \mid x_1, x_2, .., x_m)$ which f_u and f_k is a part of the same browser instance. The hybrid linking algorithm is described as follows.

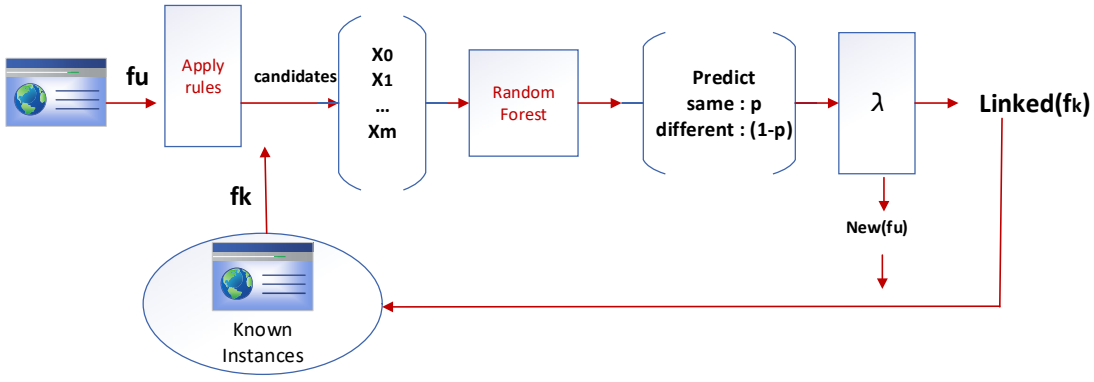


Figure H-2 the hybrid linking algorithm

Hybrid Matching Algorithm

```
function FINGERPRINTMATCHING(F, f_u,, λ)
      rules = {rule1 rule2, rule3}
      candidates ← ∅
      exact ← ∅
       F_ksub ← ∅
      for f_k ∈ F do
        if VERIFYRULES (f_k , f_u , rules) then
              if nbDiff = 0 then
                exact ← exact ∪ ⟨ f_k ⟩
              else
                F_ksub ← F_ksub ∪ ⟨ f_k ⟩
              end if
          end if
       end for
      if | exact | > 0 then
          if SAMEIDS. (exact) then
                return exact[0].id
          else
            return GENERATENEWID()
     end if
end if
candidates ← ∅
  for f_k ∈ F_ksub do
       ⟨x1,x2, .. ,xm⟩ = FEATUREVECTOR(f_u, f_k)
       p ← P (f_u.id = f_k.id | ⟨x1, x2, ... xm⟩)
      if p ≥ λ then
         candidates ← P candidates ∪ ⟨f_k,p⟩
     end if
   end for
if |candidates| > 0 then
C_h1,p_h1 ← GETCANDIDATESRANK(candidates,1)
C_h2,p_h2 ← GETCANDIDATESRANK(candidates,2)
If SAMEIDs(c_h1) and p_h1 > p_h2 + diff then
 Return candidates[0].id
end if
if SAMEIDs(c_h1 ∪ C_h2) then
    return candidates[0].id
end if
end if
return GENERATENEWID()
end function
```

GETCANDIDATESRANK considers a list of candidates and rank *i*. This function returns a list of candidates with the greatest probability.

Appendix I The process of randomization

FP-prevention breaks the fingerprint tracking by using the randomization technique to break the stability of fingerprinting technique. The FP-prevention chooses User-agent first as User-agent is associated with many attributes. Handling User-agent once can change many attributes related to a User-agent. The result of changing attributes can break the stability of fingerprint and maintains the same relationship of changed attributes.

Normally, FP-prevention collects the 100 latest versions of User-agent in a list. FP-prevention will randomly select a User-agent in a list every time of a browser's request to a web server. FP-prevention uses a concept of discrete uniform distribution (uniformly random bits) to randomize User-agent stored in a list. Therefore, each User-agent is equally likely to be randomized. It means that 100 User-agents stored in a list have equal probability 1/n = 1/100 to be randomized.

For example, if User-agent randomized by FP-prevention is

"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36".

All steps of changing attributes will be shown as follows.

1. User-agent in HTTP header field will be modified according to randomized User-agent by FP-prevention before sending a request to a web server.

2. navigator.userAgent will be changed according to randomized User-agent

3. Randomized User-agent will be split into two attributes, appVersion and appCodeName.

4. AppVersion will be split into five attributes, namely platform, oscpu, productSub, product and Vendor.

5. Plugins will be randomized according to platform.

6. mimeType will be randomized according to plugin

7. cookieEnable and JavaEnabled will be randomized true or false.

8. Screen and Font will be randomized

9. Canvas element will have noise added randomly.

The example of randomizing attributes is shown in the Figure J-1.

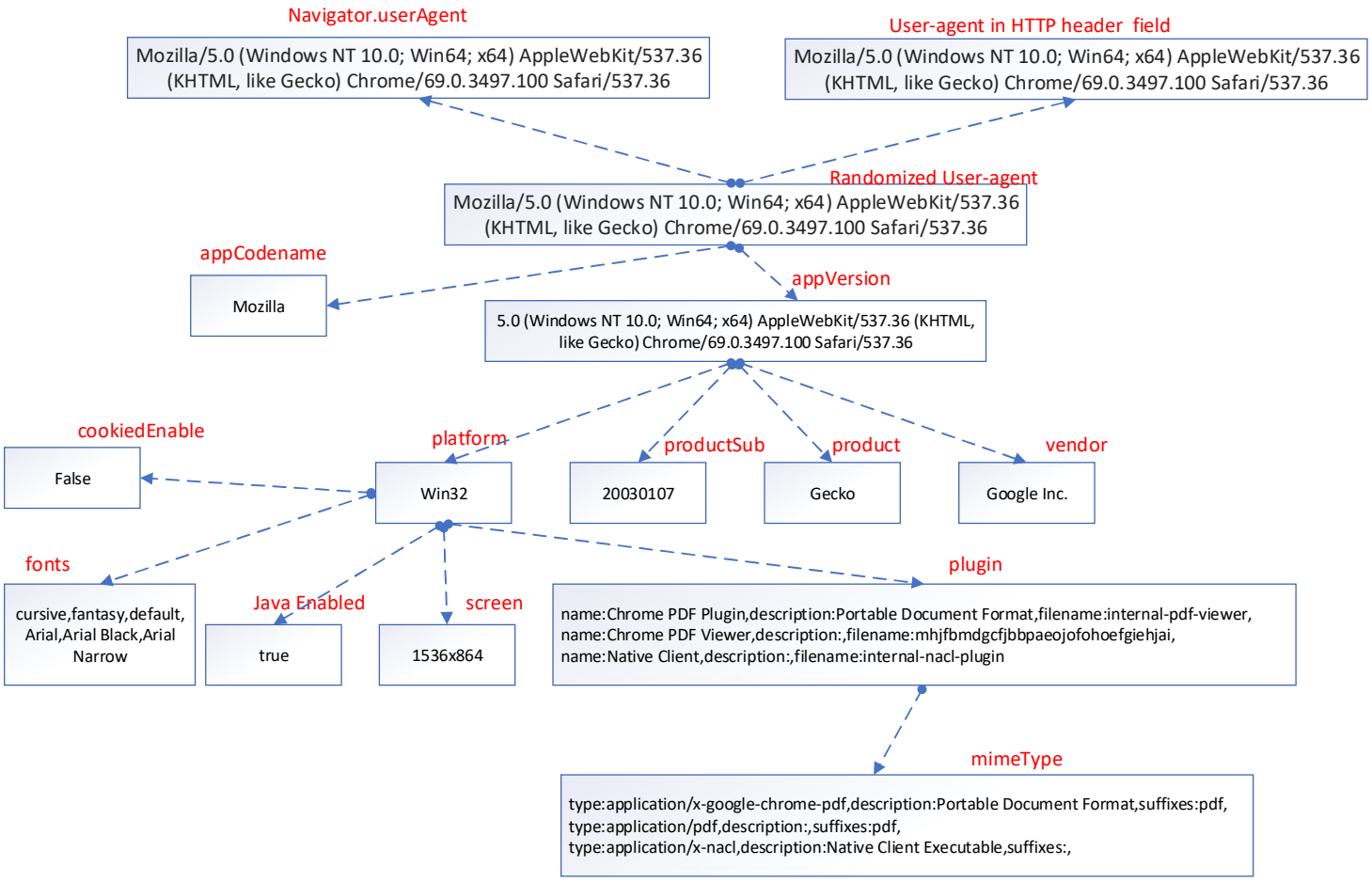


Figure J-1 the example of randomizing attributes by FP-prevention

List of References

ACAR, G. 2014. Obfuscation for and against device fingerprinting Position Paper for Symposium on Obfuscation New York University, February 15, 2014.

ACAR, G., EUBANK, C., ENGLEHARDT, S., JUAREZ, M., NARAYANAN, A. & DIAZ, C. The Web never forgets: Persistent tracking mechanisms in the wild. Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, 2014. ACM, 674-689.

ACAR, G., JUAREZ, M., NIKIFORAKIS, N., DIAZ, C., GÜRSES, S., PIESSENS, F. & PRENEEL, B. FPDetective: dusting the web for fingerprinters. Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, 2013. ACM, pp. 1129-1140.

AGGARWAL, G., BURSZTEIN, E., JACKSON, C. & BONEH, D. An Analysis of Private Browsing Modes in Modern Browsers. USENIX Security'10 Proceedings of the 19th USENIX conference on Security 2010. USENIX Association Berkeley, CA, USA ©2010 pp. 79-94.

ALACA, F. & VAN OORSCHOT, P. C. Device fingerprinting for augmenting web authentication: classification and analysis of methods. Proceedings of the 32nd Annual Conference on Computer Security Applications, 2016. ACM, 289-301.

ANDALIBI, V., CHRISTOPHE, F. & MIKKONEN, T. Analysis of Paradoxes in Fingerprint Countermeasures. Proceedings of the 21st Conference of Open Innovations Association FRUCT, University of Helsinki, Helsinki, Finland, 2017. FRUCT Oy.

ASAY, C. D. 2012. Consumer Information Privacy and the Problems (s) of Third-Party Disclosures. *Nw. J. Tech. & Intell. Prop.,* 11**,** xxxi.

BAL, A. & ACHARYA, A. Biometric authentication and tracking system for online examination system. Recent Trends in Information Systems (ReTIS), 2011 International Conference on, 2011. IEEE, 209-213.

BALEBAKO, R., LEON, P., SHAY, R., UR, B., WANG, Y. & CRANOR, L. Measuring the effectiveness of privacy tools for limiting behavioral advertising. WEB 2.0 SECURITY & PRIVACY 2012, 2012. Web, 10 pages.

BARTH, A., FELT, A. P., SAXENA, P. & BOODMAN, A. Protecting Browsers from Extension Vulnerabilities. NDSS 2010, 2010 InProceedings of the 17th Network and Distributed System Security Symposium. Citeseer, pp. 1-12.

BAU, J., MAYER, J., PASKOV, H. & MITCHELL, J. C. 2013. A promising direction for web tracking countermeasures. *Proceedings of W2SP*.

BESSON, F., BIELOVA, N. & JENSEN, T. Browser randomisation against fingerprinting: A quantitative information flow approach. Nordic Conference on Secure IT Systems, 2014. Springer, 181-196.

BIELOVA, N. 2013. Survey on JavaScript security policies and their enforcement mechanisms in a web browser. *The Journal of Logic and Algebraic Programming,* 82**,** 243-262.

BODA, K. 2014. *Firegloves* [Online]. Available: http://fingerprint.pet-portal.eu/?menu=6 [Accessed 23/04/2015].

BODA, K., FÖLDES, Á. M., GULYÁS, G. G. & IMRE, S. 2012a. User tracking on the web via cross-browser fingerprinting. *Information Security Technology for Applications.* Springer.

BODA, K., FÖLDES, Á. M., GULYÁS, G. G. & IMRE, S. 2012b. User tracking on the web via cross-browser fingerprinting. *Information Security Technology for Applications,* 7161**,** pp. 31-46.

BONFIGLIO, D., MELLIA, M., MEO, M., ROSSI, D. & TOFANELLI, P. 2007. Revealing skype traffic: when randomness plays with you. *ACM SIGCOMM Computer Communication Review,* 37**,** pp. 37-48.

BROENINK, R. Using Browser Properties for Fingerprinting Purposes. 16th biennial Twente Student Conference on IT, Enschede, Holanda, 2012a.

BROENINK, R. Using Browser Properties for Fingerprinting Purposes. 16th biennial Twente Student Conference on IT, Enschede, Holanda, 2012b. 8 pages.

CALDERA, J., HAIN, J. M. & SHERLOCK, K. 2016. ENHANCED AUTOMATED ANTI-FRAUD AND ANTI-MONEY-LAUNDERING PAYMENT SYSTEM. US Patent 20,160,071,108.

CHAIRUNNANDA, P., PHAM, N. & HENGARTNER, U. Privacy: Gone with the typing! identifying web users by their typing patterns. Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on, 2011. IEEE, 974-980.

COHEN, J. 1988. Statistical power analysis for the behavioral sciences 2nd edn. Erlbaum Associates, Hillsdale.

COVA, M., KRUEGEL, C. & VIGNA, G. Detection and analysis of drive-by-download attacks and malicious JavaScript code. Proceedings of the 19th international conference on World wide web, 2010. ACM, 281-290.

DANEZIS, G. & CLAYTON, R. 2007. *Introducing traffic analysis*, Auerbach Publications, Boca Raton, FL.

DONALDSON, D. 2008. *Online Advertising History.* Masters Dissertation, London: Bournemouth Media School.

DOTY, N. 2016. Mitigating browser fingerprinting in web specifications.

ECKERSLEY, P. How unique is your web browser? PETS'10 Proceedings of the 10th international conference on Privacy enhancing technologies 2010 Springer-Verlag Berlin, Heidelberg. Springer, pp. 1-18.

ECMA, T. 1999. *TG1. ECMAScript language specification. Standard ECMA-262, 3rd*.

EGELE, M., WURZINGER, P., KRUEGEL, C. & KIRDA, E. Defending browsers against drive-by downloads: Mitigating heap-spraying code injection attacks. Detection of Intrusions and Malware, and Vulnerability Assessment, 2009. Springer, pages 88-106.

ENGLEHARDT, S. & NARAYANAN, A. Online tracking: A 1-million-site measurement and analysis. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016a. ACM, 1388-1401.

ENGLEHARDT, S. & NARAYANAN, A. 2016b. Online tracking: A 1-million-site measurement and analysis Draft: May 18, 2016. *ACM CCS 2016***,** 20 pages.

EUBANK, C., MELARA, M., PEREZ-BOTERO, D. & NARAYANAN, A. Shining the floodlights on mobile web tracking-a privacy survey. Proceedings of the IEEE Workshop on Web, 2013. 9 pages.

EVANS, D. S. The online advertising industry: Economics, evolution, and privacy. Journal of Economic Perspectives, Forthcoming, 2009. pp. 37-60.

FAIZKHADEMI, A., ZULKERNINE, M. & WELDEMARIAM, K. FPGuard: Detection and Prevention of Browser Fingerprinting. Data and Applications Security and Privacy XXIX, 2015. Springer, pp. 293-308.

FAUL, F., ERDFELDER, E., BUCHNER, A. & LANG, A.-G. 2009. Statistical power analyses using G* Power 3.1: Tests for correlation and regression analyses. *Behavior research methods,* 41**,** 1149-1160.

FEDERRATH, H. Privacy enhanced technologies: methods–markets–misuse. International Conference on Trust, Privacy and Security in Digital Business, 2005. Springer, 1-9.

FENG, Y.-C., HUANG, Y.-C. & MA, X.-M. 2017. The application of Student's t-test in internal quality control of clinical laboratory. *Frontiers in Laboratory Medicine,* 1**,** 125-128.

FIFIELD, D. & EGELMAN, S. Fingerprinting web users through font metrics. In: Proceedings of the 19th international conference on Financial Cryptography and Data Security., 2015. 10 pages.

FIORE, U., CASTIGLIONE, A., DE SANTIS, A. & PALMIERI, F. Countering Browser Fingerprinting Techniques: Constructing a Fake Profile with Google Chrome. Network-Based Information Systems (NBiS), 2014 17th International Conference on, 2014. IEEE, pp. 355-360.

FLOOD, E. & KARLSSON, J. 2012. Browser fingerprinting.

FOURIE, I. & BOTHMA, T. Information seeking: an overview of web tracking and the criteria for tracking software. Aslib Proceedings, 2007. Emerald Group Publishing Limited, 264-284.

GAL, A., EICH, B., SHAVER, M., ANDERSON, D., MANDELIN, D., HAGHIGHAT, M. R., KAPLAN, B., HOARE, G., ZBARSKY, B. & ORENDORFF, J. 2009. Trace-based just-in-time type specialization for dynamic languages. *ACM Sigplan Notices,* 44**,** 465-478.

GÓMEZ-BOIX, A., LAPERDRIX, P. & BAUDRY, B. Hiding in the Crowd: an Analysis of the Effectiveness of Browser Fingerprinting at Large Scale. WWW 2018: The 2018 Web Conference, 2018.

GONG, X., KIYAVASH, N. & BORISOV, N. Fingerprinting websites using remote traffic analysis. Proceedings of the 17th ACM conference on Computer and communications security, 2010. ACM, 684-686.

HASSENZAHL, M. User experience (UX): towards an experiential perspective on product quality. Proceedings of the 20th Conference on l'Interaction Homme-Machine, 2008. ACM, 11-15.

HELSLOOT, L. J., TILLEM, G. & ERKIN, Z. 2017. Privacy concerns and protection measures in online behavioural advertising.

HONG, L. & JAIN, A. 1998. Integrating faces and fingerprints for personal identification. *IEEE transactions on pattern analysis and machine intelligence,* 20**,** 1295-1307.

HUANG, M., ALI, R. & LIAO, J. 2017. The effect of user experience in online games on word of mouth: A pleasure-arousal-dominance (PAD) model perspective. *Computers in Human Behavior,* 75**,** 329-338.

HUPPERICH, T., MAIORCA, D., KÜHRER, M., HOLZ, T. & GIACINTO, G. On the Robustness of Mobile Device Fingerprinting: Can Mobile Users Escape Modern Web-Tracking Mechanisms? Proceedings of the 31st Annual Computer Security Applications Conference, 2015. ACM, pp. 191-200.

ISENOR, D. & ZAKY, S. G. 1986. Fingerprint identification using graph matching. *Pattern Recognition,* 19**,** 113-122.

JANG, D., JHALA, R., LERNER, S. & SHACHAM, H. An empirical study of privacy-violating information flows in JavaScript web applications. Proceedings of the 17th ACM conference on Computer and communications security, 2010. ACM, pp. 270-283.

JÄRVINEN, J. & KARJALUOTO, H. 2015. The use of Web analytics for digital marketing performance measurement. *Industrial Marketing Management,* 50**,** 117-127.

JETTER, C. & GERKEN, J. A simplified model of user experience for practical application. 2nd COST294-MAUSE, 2007. 106-111.

JOHNSON, J. P. 2013. Targeted advertising and advertising avoidance. *The RAND Journal of Economics,* 44**,** 128-144.

KENT, M. L., CARR, B. J., HUSTED, R. A. & POP, R. A. 2011. Learning web analytics: A tool for strategic communication. *Public Relations Review,* 37**,** pp. 536-543.

KHATTAK, S., FIFIELD, D., AFROZ, S., JAVED, M., SUNDARESAN, S., PAXSON, V., MURDOCH, S. J. & MCCOY, D. Do you see what i see? differential treatment of anonymous users. Network and Distributed System Security Symposium, 2016.

KOHNO, T., BROIDO, A. & CLAFFY, K. C. 2005. Remote physical device fingerprinting. *Dependable and Secure Computing, IEEE Transactions on,* 2**,** pp. 93-108.

KRISHNAMURTHY, B. & WILLS, C. E. Generating a privacy footprint on the internet. Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, 2006. ACM, pp. 65-70.

KUMAR, S. & PHROMMATHED, P. 2005. *Research methodology*, Springer.

LAMBRECHT, A. & TUCKER, C. 2013. When does retargeting work? Information specificity in online advertising. *Journal of Marketing Research,* 50**,** pp. 561-576.

LAPERDRIX, P., BAUDRY, B. & MISHRA, V. FPRandom: Randomizing core browser objects to break advanced device fingerprinting techniques. International Symposium on Engineering Secure Software and Systems, 2017. Springer, 97-114.

LAPERDRIX, P., RUDAMETKIN, W. & BAUDRY, B. Mitigating browser fingerprint tracking: multi-level reconfiguration and diversification. Proceedings of the International Symposium on Software Engineering for Adaptive and SelfManaging Systems (SEAMS'15), 2015. pp. 98-108.

LAW, E. L.-C., ROTO, V., HASSENZAHL, M., VERMEEREN, A. P. & KORT, J. Understanding, scoping and defining user experience: a survey approach. Proceedings of the SIGCHI conference on human factors in computing systems, 2009. ACM, 719-728.

LENHARD, J., LOESING, K. & WIRTZ, G. Performance measurements of Tor hidden services in low-bandwidth access networks. Applied Cryptography and Network Security, 2009. Springer, pp. 324-341.

LIMAYE, M. G. 2009. *Software testing*, Tata McGraw-Hill Education.

LU, L., CHANG, E.-C. & CHAN, M. C. Website fingerprinting and identification using ordered feature sequences. In Proceedings of the European Symposium on Research in Computer Security, 2010. Springer, pp. 199-214.

MALANDRINO, D. & SCARANO, V. 2013. Privacy leakage on the Web: Diffusion and countermeasures. *Computer Networks,* 57**,** 2833-2855.

MAYER, J. R. 2009. Any person... a pamphleteer": Internet Anonymity in the Age of Web 2.0. *Undergraduate Senior Thesis, Princeton University*.

MAYER, J. R. & MITCHELL, J. C. Third-party web tracking: Policy and technology. Security and Privacy (SP), 2012 IEEE Symposium on, 2012. IEEE, pp. 413-427.

MCGRATH, M. 2013. JavaScript in easy steps: Create functions for the web. 5 ed.: In Easy Steps.

MOWERY, K., BOGENREIF, D., YILEK, S. & SHACHAM, H. 2011a. Fingerprinting information in JavaScript implementations. *Proceedings of W2SP,* 2**,** 11 pages.

MOWERY, K., BOGENREIF, D., YILEK, S. & SHACHAM, H. 2011b. Fingerprinting information in JavaScript implementations. *in Proceedings of W2SP 2011,* 2**,** pp. 1-11.

MOWERY, K. & SHACHAM, H. 2012. Pixel perfect: Fingerprinting canvas in HTML5. *Proceedings of W2SP 2012 IEEE Computerc Society***,** pp. 1-12.

MU, R. & HU, T. 2012. Method for fingerprinting and identifying internet users. Google Patents.

MULAZZANI, M., RESCHL, P., HUBER, M., LEITHNER, M., SCHRITTWIESER, S., WEIPPL, E. & WIEN, F. Fast and reliable browser identification with javascript engine fingerprinting. Web 2.0 Workshop on Security and Privacy (W2SP), May 2013 2013. 10 pages.

NAKIBLY, G., SHELEF, G. & YUDILEVICH, S. 2015. Hardware Fingerprinting Using HTML5. *arXiv e-print arXiv:1503.01408 by Cornell University, 5 pages.*

NAMBISAN, P. & WATT, J. H. 2011. Managing customer experiences in online product communities. *Journal of Business Research,* 64**,** 889-895.

NIKIFORAKIS, N. & ACAR, G. 2014. Browser Fingerprinting and the Online-Tracking Arms Race. *IEEE Spectrum*.

NIKIFORAKIS, N., ACAR, G. & SAELINGER, D. Browse at your own risk. IEEE Spectrum Magazine, 2014 2014a North American. IEEE Journals & Magazines, pp. 30-35.

NIKIFORAKIS, N., INVERNIZZI, L., KAPRAVELOS, A., VAN ACKER, S., JOOSEN, W., KRUEGEL, C., PIESSENS, F. & VIGNA, G. You are what you include: large-scale evaluation of remote javascript inclusions. Proceedings of the 2012 ACM conference on Computer and communications security, 2012. ACM, pp. 736-747.

NIKIFORAKIS, N., JOOSEN, W. & LIVSHITS, B. 2014b. *Privaricator: Deceiving fingerprinters with little white lies* [Online]. MSR-TR-2014-26: Technical report Microsoft Corporation. Available: http://research.microsoft.com/pubs/209989/tr1.pdf [Accessed 25/06/2015].

NIKIFORAKIS, N., KAPRAVELOS, A., JOOSEN, W., KRUEGEL, C., PIESSENS, F. & VIGNA, G. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. Security and privacy (SP), 2013 IEEE symposium on, 2013. IEEE, pp. 541-555.

NIKIFORAKIS, N., KAPRAVELOS, A., JOOSEN, W., KRUEGEL, C., PIESSENS, F. & VIGNA, G. On the Workings and Current Practices of Web-Based Device Fingerprinting. Security & Privacy, IEEE, 2014c. pp. 28-36.

OFFERMANN, P., LEVINA, O., SCHÖNHERR, M. & BUB, U. Outline of a design science research process. Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology, 2009. ACM, 7.

OLEJNIK, Ł., ACAR, G., CASTELLUCCIA, C. & DIAZ, C. 2015. The leaking battery. *Data Privacy Management, and Security Assurance.* Springer.

OLEJNIK, L., CASTELLUCCIA, C. & JANC, A. Why johnny can't browse in peace: On the uniqueness of web browsing history patterns. 5th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2012), 2012. 16 pages.

ORR, C. R., CHAUHAN, A., GUPTA, M., FRISZ, C. J. & DUNN, C. W. An approach for identifying JavaScript-loaded advertisements through static program analysis. Proceedings of the 2012 ACM workshop on Privacy in the electronic society, 2012. ACM, 1-12.

PEFFERS, K., TUUNANEN, T., ROTHENBERGER, M. A. & CHATTERJEE, S. 2007. A design science research methodology for information systems research. *Journal of management information systems,* 24**,** 45-77.

PEREIRA, H. G., DE FÁTIMA SALGUEIRO, M. & RITA, P. 2016. Online purchase determinants of loyalty: The mediating effect of satisfaction in tourism. *Journal of Retailing and Consumer Services,* 30**,** 279-291.

PHIPPEN, A., SHEPPARD, L. & FURNELL, S. A practical evaluation of Web analytics. Internet Research, 2004. pp. 284-293.

PIAO, C., LI, X., PAN, X. & ZHANG, C. 2016. User privacy protection for a mobile commerce alliance. *Electronic Commerce Research and Applications,* 18**,** 58-70.

RASTOGI, E. & KHAN, M. 2015. An analytical study of online advertising and its co-relationship with green marketing for facilitating sustainable marketing effectiveness. *International Journal of Advanced Research (IJAR),* 1**,** pp. 182-184.

RATANAWORABHAN, P., LIVSHITS, B. & ZORN, B. G. 2010. JSMeter: Comparing the Behavior of JavaScript Benchmarks with Real Web Applications. *WebApps,* 10**,** 3-3.

RICHARDS, G., LEBRESNE, S., BURG, B. & VITEK, J. An analysis of the dynamic behavior of JavaScript programs. ACM Sigplan Notices, 2010. ACM, pp. 1-12.

ROESNER, F., KOHNO, T. & WETHERALL, D. Detecting and defending against third-party tracking on the web. Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, 2012. USENIX Association, 12-12.

SAUNDERS, M., LEWIS, P. & THORNHILL, A. 1997. Collecting primary data using questionnaires. *Research methods for business students***,** 354-405.

SCHULZE, K. & KRÖMKER, H. A framework to measure user experience of interactive online products. Proceedings of the 7th International Conference on Methods and Techniques in Behavioral Research, 2010. ACM, 14.

SHAHRIAR, H., WELDEMARIAM, K., ZULKERNINE, M. & LUTELLIER, T. 2014. Effective detection of vulnerable and malicious browser extensions. *Computers & Security,* 47**,** 66-84.

SINHA, R. & CHOUDHARY, C. 2014. Information Leak Detection System using Fingerprint of data. *International Journal on Recent and Innovation Trends in Computing and Communication* 2**,** pp. 3911-3915.

SONG, D. X., WAGNER, D. & TIAN, X. Timing Analysis of Keystrokes and Timing Attacks on SSH. USENIX Security Symposium, 2001. 17 pages.

SORENSEN, O. Zombie-cookies: Case studies and mitigation. Internet Technology and Secured Transactions (ICITST), 2013 8th International Conference for, 2013. IEEE, pp. 321-326.

STAROV, O. & NIKIFORAKIS, N. Extended tracking powers: Measuring the privacy diffusion enabled by browser extensions. Proceedings of the 26th International Conference on World Wide Web, 2017. International World Wide Web Conferences Steering Committee, 1481-1490.

STUDENT 1908. The probable error of a mean. *Biometrika***,** 1-25.

TORRES, C. F., JONKER, H. & MAUW, S. FP-Block: Usable Web Privacy by Controlling Browser Fingerprinting. Computer Security--ESORICS 2015, 2015. Springer, pp. 3-19.

TRAN, M., DONG, X., LIANG, Z. & JIANG, X. Tracking the trackers: Fast and scalable dynamic analysis of web content for privacy violations. Applied Cryptography and Network Security, 2012. Springer, 418-435.

TSCHOFENIG, H. & VAN EIJK, R. 2011. *DO NOT TRACK* [Online]. w3.org Available: http://www.w3.org/2011/track-privacy/papers/Tschofenig.pdf [Accessed 21/06/2015].

TUROW, J., KING, J., HOOFNAGLE, C. J., BLEAKLEY, A. & HENNESSY, M. 2009. Americans reject tailored advertising and three activities that enable it. *Available at SSRN 1478214*.

UPATHILAKE, R., LI, Y. & MATRAWY, A. A classification of web browser fingerprinting techniques. New Technologies, Mobility and Security (NTMS), 2015 7th International Conference on, 2015. IEEE, pp 1-5.

VANITHA, S. & ARTHIBALA, U. P. A. Third-Party Net Stalk: Policy and Technology. *International Journal of Computer Science and Information Technologies (IJCSIT),* 5(6)**,** pp. 7480-7487.

VOS, P., HOGERS, R., BLEEKER, M., REIJANS, M., VAN DE LEE, T., HORNES, M., FRITERS, A., POT, J., PALEMAN, J. & KUIPER, M. 1995. AFLP: a new technique for DNA fingerprinting. *Nucleic acids research,* 23**,** 4407-4414.

WADKAR, H., MISHRA, A. & DIXIT, A. Prevention of information leakages in a web browser by monitoring system calls. Advance Computing Conference (IACC), 2014 IEEE International, 2014. IEEE, pp. 199-204.

WELDEMARIAM, K. FPGuard: Detection and Prevention of Browser Fingerprinting. Data and Applications Security and Privacy XXIX: 29th Annual IFIP WG 11.3 Working Conference, DBSec 2015, Fairfax, VA, USA, July 13-15, 2015, Proceedings, 2015. Springer, 293.

WINDRUM, P. 2004. Leveraging technological externalities in complex technologies: Microsoft's exploitation of standards in the browser wars. *Research Policy,* 33**,** 385-394.

YEN, T.-F., XIE, Y., YU, F., YU, R. P. & ABADI, M. Host Fingerprinting and Tracking on the Web: Privacy and Security Implications. In Proceddings of the 19th nnual Network and Distributed System Security Symposium (NDSS), 2012a. 16 pages.

YEN, T.-F., XIE, Y., YU, F., YU, R. P. & ABADI, M. Host Fingerprinting and Tracking on the Web: Privacy and Security Implications. NDSS, 2012b. 16 pages.

ZUO, L., WANG, Y. & TAN, T. Personal handwriting identification based on pca. Second International Conference on Image and Graphics, 2002. International Society for Optics and Photonics, 766-771.