JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 34 (2025) 60–67
Received: 14 October 2024
Accepted: 11 November 2024

Analysis of User Identification Methods in Web Browsers

Analiza metod identyfikacji użytkowników w przeglądarkach internetowych

Eduard Chyzhik*, Jakub Smołka

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract
Website visitor data helps to increase application security, control limited access, personalize content, collect data and improve the application. The article analyzes user identification methods in a browser: Browser Fingerprint, Supercookie, Keystroke Dynamics. The analysis of the methods was perfomed based on 4 criteria: False Acceptance Rate, applicability, avoidance resistance, Frontend Performance. For the research, literature, tests in different browsers, demo scripts in JavaScript, and the Performance API were used. It has been shown that the Supercookie is ineffective today, but Browser Fingerprint and Keystroke Dynamics can still be utilized for identification.

Keywords: user identification; Browser Fingerprint; Zombie Cookies; Biometric Fingerprint

Streszczenie
Dane o odwiedzających stronę internetową pomagają zwiększyć bezpieczeństwo aplikacji, kontrolować ograniczony dostęp, personalizować treści, zbierać dane oraz ulepszać aplikację. Artykuł analizuje metody identyfikacji użytkowników w przeglądarce: odcisk przeglądarki, superciasteczko, dynamikę naciśnięć klawiszy. Analiza metod została przeprowadzona na podstawie 4 kryteriów: wskaźnika błędnej akceptacji, zastosowalności, odporności na unikanie, wydajności frontendu. Do badań wykorzystano literaturę, testy w różnych przeglądarkach, skrypty demonstracyjne w JavaScript oraz Performance API. Wykazano, że Supercookie jest dziś nieskuteczny, ale Browser Fingerprint i Keystroke Dynamics mogą nadal być wykorzystywane do identyfikacji.

Słowa kluczowe: identyfikacja użytkownika; odcisk palca przeglądarki; nieusuwalne ciasteczka; biometryczny odcisk palca

*Corresponding author
Email address: s101292@pollub.edu.pl (E. Chyzhik)

1. Introduction

The world stopped to be anonymous long time ago: every person has a passport, their biometric data is stored in databases, there are cameras on the streets that recognize faces. But the Internet is a new virtual world that has brought countless opportunities, including anonymity. The active confrontation between anonymity and identification appeared in the 1990s. When the Internet began to develop rapidly and the number of users began to grow incredibly quickly, companies were puzzled by the ways of identifying users. As a result, methods using IP, cookies, HTTP-sessions, basic athentications appeared. Tools and projects for anonymization have also emerged: proxy, VPN, Onion Routing (Tor), Freenet, Mixmaster.

User identification gives service owners great opportunities from additional verification to the analysis of interests and needs. With the rapid development of machine learning technologies, every byte of information collected can become an important detail in the user's analysis and prediction mechanism.

Today, many approaches to identification are no longer relevant. But thanks to the development of technologies, the expansion of browser functionality, new ones are constantly appearing.

The purpose of the article is to analyze current identification methods, evaluate them according to the specified criteria (False Acceptance Rate, applicability, avoidance resistance, Frontend Performance), and assess their prospects.

2. Literature overview

Article “Hiding in the Crowd: an Analysis of the Effectiveness of Browser Fingerprinting at Large Scale” [1] presents an analysis of Browser Fingerprinting based on a dataset of over 2 million browser fingerprints collected from one of the top 15 websites in France. The analysis finds fingerprinting less effective than earlier researches, with only 33.6% unique fingerprints compared to over 80% previously. The results are compared with Panopticlick and AmIUnique, and implications for the future of fingerprinting are discussed. Findings show that current fingerprinting techniques do not provide effective mechanisms to uniquely identify users belonging to a specific demographic region as 33.6% of collected fingerprints were unique in the dataset [1].

Research presented in “Browser Fingerprinting: A survey” [2] explores the evolution of Browser Fingerprinting, a technique that gathers device-specific information for identification and security purposes, born out of the diversity of modern web devices. It discusses how HTML5 and CSS3 enabled richer and more dynamic web experiences but also opened the door to fingerprinting techniques that track users without direct consent.

Authors discuss fingerprinting in 4 areas: science, technology, business, regulation. The article ends with the call for further research for better understanding and regulating this technique, highlighting not only its potential to improve online security but also posing risks to privacy [2]. This paper contributes valuable insights into the evolution of tracking technologies and browser-based privacy measures, showing how advancements in web standards and user privacy tools are gradually rendering Evercookie obsolete.

Jonas Strehle suggested a new approach for Immortal Cookies [3]. Modern browsers cache Favicons, small icons representing websites in a database called the Favicon cache (F-Cache). This cache records the URL and Favicon data. The tracking method exploits this by assigning unique identifiers to users based on which Favicons are requested or cached [3].

"A Systematic Literature Review on Latest Keystroke Dynamics Based Models" [4] provides a comprehensive analysis of recent advancements in Keystroke Dynamics for user authentication, identification, and prediction. Accuracy for identifying users from a group was 90.38%, improving to 95.24% with continuous patterns. Predictive models showed 91.09% accuracy in static and 81.95% in dynamic modes. The research highlights significant variability across Keystroke Dynamics designs and emphasizes the need for standardized assessment methods. It offers insights into current challenges and future research directions, aiming to advance the field of Keystroke Dynamics-based user authentication and identification [4].

3. Methods of user identification

In the research, all methods have been divided into 3 semantic categories: Browser Fingerprint [5], Zombie Cookies [6], Biometric Fingerprint. The categories of Zombie Cookies and Biometric Fingerprint are very broad. Therefore, only one method from each category will be considered: Supercookie from Zombie Cookies and Keystroke Dynamics from Biometric Fingerprint.

3.1. Browser Fingerprint

Identification methods that work on the basis of obtaining data about the user, browser, and device have been included in this category. Moreover, the more unique this information is and the less often it changes, the more unique the fingerprint will be. The information can be obtained either directly (for example, from the global navigator object) or indirectly (for example, by creating a browser- and device-specific canvas object and then encoding it in base64 for comparison). Browsers reduce the amount of information available for privacy reasons, but there is still enough information to identify the browser.

3.2. Zombie Cookies

Methods based on generating a unique identifier and attempting to record and assign it to a user in such a way that clearing it would be as difficult as possible have been included in this category.

Supercookie method is based on browser caching of the Favicon. If the icon is cached in the browser, then the request to the server is not sent, otherwise a request is sent to the server, which can be tracked and appropriate conclusions can be made. Binary representation is used to encode the identifier. 0 means that the icon has been successfully loaded from the cache, and 1 means that a request has been sent to the server. Thus, the identifier can be obtained in its bit representation thanks to the redirect chain. For example, the bit string 1111111100 corresponds to the identifier 1020.

3.3. Biometric Fingerprint

In this category, methods that allow for user identification based on unique biometric behavior (e.g., based on the duration of keystrokes and the delay between them during typing) were included.

Keystroke Dynamics [7] identification is a biometric method that analyzes the unique patterns of an individual's typing behavior on a keyboard. This technique utilizes various attributes related to the timing and rhythm of keystrokes, making it possible to verify a person's identity based on their typing characteristics.

4. Reserch method

The analysis of the methods will be conducted based on four aspects: False Acceptance Rate, applicability, avoidance resistance, and Frontend Performance.

False Acceptance Rate (FAR) [8] is a metric that defines the probability of incorrect identifications, specifically the likelihood that different users may be incorrectly recognized as the same individual. In other words, it measures the rate at which the identification system erroneously accepts an unknown user or fails to distinguish between distinct users, leading to potential collisions of identifiers. A lower FAR indicates a more accurate identification system, reducing the risk of misidentification among users.

Applicability refers to the extent to which a user identification method can be effectively utilized across different conditions. This metric evaluates how well the identification method performs in diverse scenarios, including various web sites, browsers, devices and user behaviors. An exemplary highly applicable identification method should operate in any browser, with any settings, on any operating system, on any website, and on any device.

Avoidance resistance examines how difficult it is to circumvent a specific identification method (i.e., to convince the system that one is a new user). For some methods, it may be sufficient to open a new browser or use incognito mode. The ideal method according to this metric would ensure that the identifier remains unchanged regardless of any actions taken by the user, including switching browsers, devices, or internet service providers.

Frontend Performance measures the time taken on the client side to execute the necessary actions for user identification.

4.1. Browser Fingerprint

Browser Fingerprint False Acceptance Rate

False Acceptance Rate was determined based on a literature research.

Browser Fingerprint applicability

Browser Fingerprint applicability was found based on the experience gained during the development of the demo script. To run the script, only a browser environment and enabled JavaScript are required.

Browser Fingerprint avoidance resistance

To analyze the Browser Fingerprint avoidance resistance, a module written in JavaScript was developed. This module received available information from the browser and also generated fingerprints based on various technologies. The result is returned in JSON format. The returned object has 3 properties: navigator, general, device.

Navigator consists of all the current methods of obtaining information from the navigator interface described in the specification [9]: activeVRDisplays, bluetooth, buildID, clipboard, connection, contacts, cookieEnabled, credentials, globalPrivacyControl, deviceMemory, geolocation, keyboard, language, gpu, hardwareConcurrency, hid, ink, languages, locks, login, maxTouchPoints, mediaDevices, mediaSession, onLine, pdfViewerEnabled, permissions, presentation, scheduling, serial, serviceWorker, storage, usb, userActivation, userAgent, userAgentData, virtualKeyboard, wakeLock, webdriver, windowControlsOverlay, xr, canShare, clearAppBadge, battery, deprecatedReplaceInURN, getAutoplayPolicy, gamepads, installedRelatedApps, protocolHandler, mediaKeySystemAccess, midiAccess, sendBeacon, vibrate, cpuClasses, setAppBadge, share, unregisterProtocolHandler.

General consists of general information: screenSize, availableSize, colorDepth, isWebSocketSupported, pixelRatio, mathtan, sessionStorageSupported,, dateFormat, userData, localStorageSupported, isAdBlockEnabled.

Device fingerprint has 5 properties: canvas, webgl, audio, font, media.

Device.canvas fingerprint. Unique fingerprint generated by rendering of various graphical elements on an HTML canvas. It's available because canvas fingerprint depends on a lot of parameters such as the main ones are: installed fonts, graphics card, browser rendering engine, operating system. This property contains the string (hash of the received data).

Device.webgl fingerprint. Unique WebGL-based fingerprint generated by rendering a simple graphic using WebGL and then capturing the pixel data of the rendered image. WebGL is highly dependent on the graphics hardware (GPU) and the specific drivers used by the device. Different GPUs and drivers can cause subtle differences in how the same WebGL content is rendered, even though the code is identical. For example, floating-point calculations, shading, and anti-aliasing can differ slightly across different devices. The operating system and browser rendering engines also affect WebGL rendering. Each browser and system may implement WebGL with slight variations, especially in terms of floating-point precision, texture filtering, and color accuracy. These variations influence the final image, even when rendering the same graphic with the same code. This property contains a string (hash of the received data).

Device.audio fingerprint. Audio-based fingerprint generated by using the Web Audio API to create and analyze a sound signal. The Web Audio API, which interacts closely with the system's audio stack, generates a signal that varies slightly between devices and environments, allowing this method to produce a unique audio fingerprint based on the captured frequency data. This property contains a string (hash of the received data).

Device.font fingerprint. Unique fingerprint is generated based on the fonts installed on user's device. By detecting which fonts are available on the system, this method can generate a fingerprint that is unique to the combination of fonts installed on the device. Even when the same fonts are available on two different systems, slight differences in how the fonts are rendered (due to operating system or browser-level rendering mechanisms) may lead to slight variations in size, further enhancing the uniqueness of the fingerprint. This property contains a string (hash of the received data).

Device.media fingerprint. Fingerprint based on the media decoding capabilities of the user's device. Different devices and browsers support different media formats, codecs, and decoding methods such factors as hardware (GPU, CPU), installed codecs, and the browser engine affect which formats can be decoded. Thus, by checking a broad range of formats and codecs, this method produces a fingerprint that helps identify the user's environment. This property contains a string (hash of the received data).

For the tests, a laptop with AMD Ryzen 5 4500U, Radeon Graphics, and Windows 10 22H2 with the following browsers was used:

- Chrome (112.0.5615.121 64-bit),
- Firefox (130.0.1 64-bit),
- Brave (1.69.162 Chromium 64-bit).

In addition to that, phone MediaTek Dimensity 9200, Android 11 with following browsers was used:

- Chrome (128.0.6613.128),
- Firefox (130.0),
- Brave (1.69.164 Chromium).

The following hypotheses were formulated:

H1. The same browser will always return the same fingerprints. Even after restarting the browser.

H2. Switching browser modes or changing browsers will result in alterations to browser fingerprints.

To test hypotheses H1 and H2, a script comparing JSON strings was additionally developed. It takes two JSON strings and:

- calculates the percentage of unchanged properties for general, navigator objects,
- compares the strings (hash) of properties canvas, webgl, audio, fonts, media.

To test the hypothesis H1, 10 pairs of tests were conducted (5 on Windows and 5 on Android). To do this, a fingerprint was generated in Chrome, Chrome Incognito, Firefox, Firefox Incognito, and Brave, and then each browser was restarted and the values were compared. This way, the percentage of data that remains unchanged when the browser is restarted was determined.

To test the hypothesis H2, a fingerprint was generated in the Chrome browser, and the results were recorded. Then, a fingerprint was generated in Chrome Incognito, Firefox, and Brave, and these were compared with the results obtained in Chrome. This is how the percentage of data changeability when changing modes and browsers was determined.

Browser Fingerprint Frontend Performance

To improve the performance, navigator, general and device objects were generated in parallel. To analyze the generation time of each object and all data taken together, Performance API [10] was used. Thus, the duration of generation of each object separately and all objects together were obtained. The duration is affected by the browser, operating system, device. Therefore, the tests were conducted on 2 devices and in 3 browsers.

For the tests, a laptop with AMD Ryzen 5 4500U, Radeon Graphics, and Windows 10 22H2 with the following browsers was used:

- Chrome (112.0.5615.121 64-bit),
- Firefox (130.0.1 64-bit),
- Brave (1.69.162 Chromium 64-bit).

In addition to that, phone MediaTek Dimensity 9200, Android 11 with following browsers was used:

- Chrome (128.0.6613.128),
- Firefox (130.0),
- Brave (1.69.164 Chromium).

4.2. Supercookie

Supercookie False Acceptance Rate

False Acceptance Rate was defined based on the way the method works. Collisions are impossible because a new, guaranteed unique identifier is generated for each user. An existing identifier can only be read, but regenerating an identical identifier is not possible. Therefore, False Acceptance Rate will be equal to 0%.

Supercookie applicability

Applicability was determined through the literature research and using the test environment of the Supercookie method available on the developer's page [11]. During test identification on the test environment, it was revealed that the site makes many redirects (not in the background), which significantly affects the possibility of using this method in various situations.

Supercookie avoidance resistance

To explore this aspect, a hypothesis was formulated:

H3. The method can be used for identification in incognito mode.

To test hypothesis H3, the following algorithm was developed for each browser in two modes (default and incognito):

1. Open pollub.pl and check from where Favicon was loaded (using browser Dev Tools).
2. Reload page and check from where Favicon was loaded (using browser Dev Tools).
3. Switch to incognito mode and check from where Favicon was loaded (using browser Dev Tools).

Supercookie Frontend Performance

Frontend Performance was determined based on a literature research.

4.3. Keystroke Dynamics

Keystroke Dynamics False Acceptance Rate

False Acceptance Rate was determined based on a literature research.

Keystroke Dynamics applicability

Applicability was found based on the experience gained in demo script developing and based on method technology. To apply the Keystroke Dynamics method, it is necessary for the user to input data from the keyboard. Also JavaScript enabled is required.

Keystroke Dynamics avoidance resistance

To explore this aspect, two hypotheses were formulated:

H4. Chrome, Firefox, Brave accurately detect the time of pressing and releasing keys.

H5. The keyboard connection method, device and operating system do not affect the error in the timing of pressing and releasing keys.

A JavaScript script was developed, which uses a KeyboardEvent [12] and the Performance API [10] to log Keystroke Dynamics. Keystrokes are tracked, and the duration of keypresses is recorded by this script. On the script page, the cursor is placed in the input field, and the 'q' key on the keyboard is held down to exclude any physiological errors. Then, the first 502 presses are taken from the total data set, and the first 2 are excluded. As a result, the duration of each of the 500 presses was obtained.

For each data set, the following was found:

- Median,
- Minimum,
- Maximum,
- Mean Absolute Deviation (MAD),
- Maximum deviation.

To test hypothesis H4, these tests were conducted in three browsers on one device (laptop AMD Ryzen 5 4500U, Radeon Graphics with Ubuntu 22.04):

- Chrome (112.0.5615.121 64-bit),
- Firefox (130.0.1 64-bit),
- Brave (1.69.162 Chromium 64-bit).

To test hypothesis H5, these tests were conducted with different configurations:

- On laptop №1 (AMD Ryzen 5 4500U, Radeon Graphics with Ubuntu 22.04, Chrome 112.0.5615.121 64-bit) with default keyboard,
- On laptop №1 (AMD Ryzen 5 4500U, Radeon Graphics with Ubuntu 22.04, Chrome 112.0.5615.121 64-bit) with external keyboard connected by Bluetooth,
- On laptop №2 (Intel i5-8265U, Intel UHD Graphics 620 with Windows 10 22H2, Chrome 112.0.5615.121 64-bit) with default keyboard.

Keystroke Dynamics Frontend Performance

To obtain Frontend Performance metrics, three tests were executed in three browsers:

- Type one character on the keyboard (“q”) in the Chrome, Firefox, Brave,
- Type simple password on the keyboard (“qwerty123”) in the Chrome, Firefox, Brave,
- Type comment on the keyboard (“Explore algorithms, data structures, AI, and more in the world of computer science innovation!” in the Chrome, Firefox, Brave.

The duration of logging keystrokes was tracked by the Performance API [10].

5. Result analysis

5.1. Browser Fingerprint

Browser Fingerprint False Acceptance Rate

In the context of the researched articles, the percentage of non-unique fingerprints represents the False Acceptance Rate (FAR), which refers to the rate at which browsers share identical fingerprints and, as a result, cannot be distinguished from one another.

In the study "Hiding in the Crowd: an Analysis of the Effectiveness of Browser Fingerprinting at Large Scale", researchers found that only 33.6% of browser fingerprints were unique [1]. Thus, FAR is 66.4%.

The Panopticlick experiment, created by Peter Eckersley at the Electronic Frontier Foundation, demonstrated that 83% of desktop browsers had unique fingerprints [13]. Thus, FAR is 17%.

In the study by Laperdrix and colleagues using the AmIUnique platform, 89.4% of browser fingerprints were found to be unique [14]. Thus, FAR is 10.6%.

The False Acceptance Rate depends on by many factors. For instance, the "Hiding in the Crowd" study focused on users belonging to a specific demographic region [1], which likely reduced the number of unique fingerprints. Furthermore, the amount and type of data collected, as well as ongoing browser updates and changes in supported features, also affect this metric. Based on a review of the literature, FAR in Browser Fingerprinting methods varies between 10.6% and 66.4%.

Browser Fingerprint applicability

To use this identification method, only browser environment and JavaScript enabled are required.

Browser Fingerprint avoidance resistance

Hypothesis H1 was disproved, because Firefox incognito and Brave have changed data. However, this change is not critical. It will clearly complicate the identification algorithm, but is unlikely to prevent identification (Table 1 and Table 2).

Table 1: Percentage of unchanged data when restarting the browser (Android). B1 − Chrome browser, B2 − Firefox browser, B3 − Brave browser. Each of these designations may have “/i” added, indicating the use of incognito mode

	B1 (%)	B1/i (%)	B2 (%)	B2/i (%)	B3 (%)
General	100	100	100	100	100
Navigator	100	100	100	100	98
Device	100	100	100	60	100

Table 2: Percentage of unchanged data when restarting the browser (Windows). B1 − Chrome browser, B2 − Firefox browser, B3 − Brave browser. Each of these designations may have “/i” added, indicating the use of incognito mode

	B1 (%)	B1/i (%)	B2 (%)	B2/i (%)	B3 (%)
General	100	100	100	100	100
Navigator	100	100	100	100	98
Device	100	100	100	60	100

Table 3: Percentage of unchanged data when changing the browser/mode (Android). B1 − Chrome browser, B2 − Firefox browser, B3 − Brave browser. Each of these designations may have “/i” added, indicating the use of incognito mode

	B1 (%)	B1/i (%)	B2 (%)	B3 (%)
General	100	100	100	100
Navigator	100	96	42	84
Canvas	100	100	0	0
Webgl	100	100	100	100
Audio	100	100	100	100
Fonts	100	100	0	100
Media	100	100	0	100

Table 4: Percentage of unchanged data when changing the browser/mode (Windows). B1 − Chrome browser, B2 − Firefox browser, B3 − Brave browser. Each of these designations may have “/i” added, indicating the use of incognito mode

	B1 (%)	B1/i (%)	B2 (%)	B3 (%)
General	100	100	100	83
Navigator	100	98	47	77
Canvas	100	100	0	0
Webgl	100	100	100	100
Audio	100	100	100	100
Fonts	100	100	0	0
Media	100	100	0	0

Hypothesis H2 was confirmed. Only Chrome Incognito has minor differences with Chrome. Firefox and Brave have large differences, which will prevent identification (Table 3 and Table 4).

Browser Fingerprint Frontend Performance

Performance tests showed promising results, especially considering that they can be performed in the background while the main content is loaded (Table 5 and Table 6).

Table 5: Browser Fingerprint Frontend Performance (Windows). B1 − Chrome browser, B2 − Firefox browser, B3 − Brave browser

	B1 (ms)	B2 (ms)	B3 (ms)
General	0,19	0,19	0,16
Navigator	67,61	56,71	50,03
Device	61,84	53,49	46,44
Total	67,92	58,84	50,23

Table 6: Browser Fingerprint Frontend Performance (Android). B1 − Chrome browser, B2 − Firefox browser, B3 − Brave browser

	B1 (ms)	B2 (ms)	B3 (ms)
General	0,60	0,65	0,77
Navigator	195,74	191,43	202,71
Device	176,96	162,98	183,31
Total	196,64	193,22	204,08

Browser Fingerprint conclusions

The False Acceptance Rate depends on various factors: the similarity of identified users (e.g. by demographic region), the data collected, the algorithm for analyzing the collected data. In the studies researched, FAR ranges from 10.6% to 66.4%.

Applicability turned out to be extremely high. After all, this method is not applicable only to browsers in which JavaScript is disabled.

Avoidance resistance has shown that today the method is able to effectively identify the user within one browser (even when switching between normal and incognito modes). However, it is useless when changing browsers and will identify the user as a new one.

Frontend Performance has shown excellent results and confirmed that the method can be applied effectively.

5.2. Supercookie

Supercookie False Acceptance Rate

False Acceptance Rate is 0%. This makes this method very reliable with additional verifications.

Supercookie applicability

For this method, JavaScript does not need to be enabled, but the user needs to be redirected from address to address more than 20 times to achieve a sufficient range of possible identifiers. This behavior resembles phishing or spamming and is likely to scare users.

Supercookie avoidance resistance

As shown in Table 7 this method of identification is ineffective in incognito modes because each a new session has clear cache (including Favicon cache) and cookies. Therefore, restarting the incognito mode is enough to get a new identifier. Hypothesis H3 is disproved. The method is applicable only in normal browser mode and within one browser. When another browser or incognito mode is switched to, a new identifier will be generated.

Table 7: Cache persistence. B1 − Chrome browser, B2 − Firefox browser, B3 − Brave browser. Each of these designations may have "/i" added, indicating the use of incognito mode

	B1	B1/i	B2	B2/i	B3	B3/i
Cache is saved after restart	+	-	+	-	+	-

Supercookie Frontend Performance

Depends on redirects count (count of available unique identifiers). One redirect can accommodate 1 bit of data. An identifier encoded in any number of bits can be obtained by using a chain of redirects. To calculate how many unique identifiers can be encoded using *N* bits (redirects), formula 1 should be applied. For example, to encode up to 1,024 unique identifiers, 10 bits or 10 redirects are needed for reading and 10 redirects are needed for writing. This amount is suitable for a service with a very small number of users. Reading 10 bits via redirects takes about 1 second, while writing takes about 1.2 seconds [3]. In total, this results in approximately 2.2 seconds. To identify up to 4,294,967,296 unique users, 32 bits or 32 redirects for reading and 32 redirects for writing are required. Reading 32 bits will take about 3 seconds, and writing will take another 3 seconds [3], for a total of 6 seconds. This is a considerable amount of time, especially considering that the redirects are not performed in the background but rather directly within the browser tab.

$$2^N = U \tag{1}$$

where *N* is the number of redirects, and *U* is the maximum number of unique identifiers.

Supercookie conclusions

The method can be used in situations where an False Acceptance is unacceptable. However, this method is not effective in incognito, when changing browsers, and also has a very long execution time on the client side.

5.3. Keystroke Dynamics

Keystroke Dynamics False Acceptance Rate

The study by E. Alamri, A. Alnajim, and S. Alsuhibany, "Investigation of Using CAPTCHA Keystroke Dynamics to Enhance the Prevention of Phishing Attacks", published in Future Internet (2022), reported a False Acceptance Rate (FAR) of 17% for CAPTCHA-based Keystroke Dynamics [15].

The study "Free-text Keystroke Dynamics Authentication for Arabic Language" by A. Alsultan, K. Warwick, and H. Wei, published in IET Biometrics (2016), showed that the False Acceptance Rate (FAR) for Arabic language free-text Keystroke Dynamics authentication ranged from 24.5% to 28.1% [16].

Therefore, based on literature researching False Acceptance Rate is between 17% and 28.1%.

Keystroke Dynamics applicability

This method requires JavaScript to be enabled, as well as context in which the user is entering text, such as completing a HTML form.

Keystroke Dynamics avoidance resistance

Hypothesis H4 is disproved. Firefox and Brave are rounding results to whole numbers. This makes result less accurate. Also each browser has different results (Table 8 and Figure 1), which makes the identification of the same user in different browsers more difficult.

Table 8: Keystroke Dynamics in different browsers. B1 − Chrome browser, B2 − Firefox browser, B3 − Brave browser

	B1 (ms)	B2 (ms)	B3 (ms)
MAD	0,77	0,78	0,86
Median	30,30	30,00	30,00
Minimum	7,10	27,00	21,00
Maximum	53,70	34,00	43,00
Maximum deviation	46,60	7,00	22,00

Table 9: Keystroke Dynamics on different OSes and keyboards. 1 - laptop №1 (Ubuntu 22.04, Chrome) with default keyboard, 2 - laptop №1 (Ubuntu 22.04, Chrome) with external keyboard connected by Bluetooth, 3 - laptop №2 (Windows 10 22H2, Chrome) with default keyboard

	1 (ms)	2 (ms)	3 (ms)
MAD	0,68	0,53	5,75
Median	30,30	30,40	30,20
Minimum	1,00	23,60	0,60
Maximum	61,30	38,70	91,80
Maximum deviation	60,30	15,10	91,20

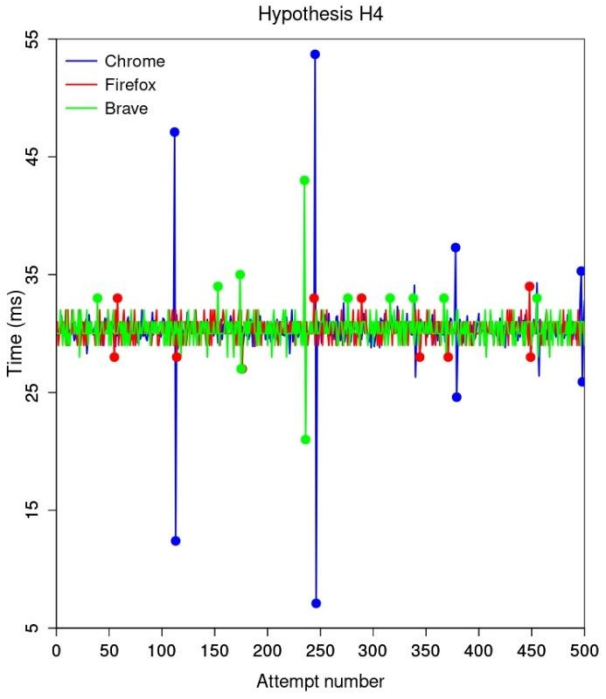

Figure 1: The duration of a single key press across different browsers.

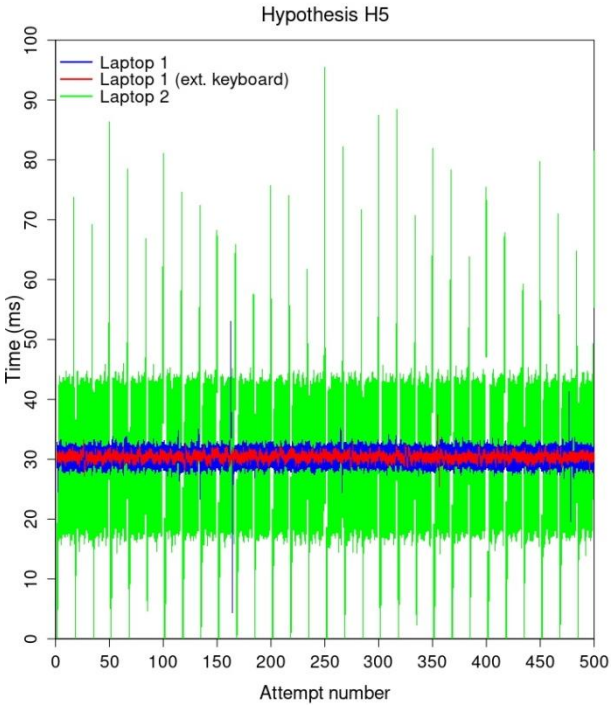

Figure 2: The duration of a single key press, with a one standard deviation (±1 SD) range, measured across different operating systems and keyboards

Hypothesis H5 is disproved. Different devices, different keyboards, different operating systems affect the data and distort it (

Table 9 and Figure 2). The only maximally similar value is median. It will complicate the identification of the same user with different keyboard or operation system.

Keystroke Dynamics Frontend Performance

To obtain Frontend Performance metrics, 3 tests were executed in 3 browsers:

- Type one character on the keyboard (“q”). Chrome, Firefox, Brave.
- Type simple password on the keyboard (“qwety123”). Chrome, Firefox, Brave.
- Type comment on the keyboard (“Explore algorithms, data structures, AI, and more in the world of computer science innovation!”). Chrome, Firefox, Brave.

According to the test data, the duration of logging button presses is negligible (Table 10).

Table 10: Keystroke Dynamics Frontend Performance. B1 − Chrome browser, B2 − Firefox browser, B3 − Brave browser

	B1 (ms)	B2 (ms)	B3 (ms)
One symbol	0,1	1,0	0,0
Simple password	1,3	1,0	1,0
Comment 100 symbols	9,5	10,0	6,0

Keystroke Dynamics conclusions

This method is valuable because it is not technically tied to a browser or device. However, as experiments demonstrated, the browser, device, and operating system affect the results obtained. Knowing the user's browser, there is a chance to take these distortions into account and predict them. However, it is unlikely that the distortions of the device and keyboard will be predicted. Therefore, this method is probably most effective in one system, device, and, preferably, one browser.

6. Conclusions

Based on the research conducted, it can be concluded that each of the methods considered has its own advantages and disadvantages.

The Supercookie method has proven to be completely ineffective today due to its long, explicit execution directly in the browser tab and low avoidance resistance (to become a new user, it is enough to switch to incognito mode or clear the cache). This method has only 2 obvious advantages: no False Acceptance and no need for enabled JavaScript.

The Browser Fingerprint and Keystroke Dynamics methods are highly dependent on the implementation of the algorithms used to analyze the collected data. The data collection process on the frontend is fast, and due to their applicability and resistance to evasion, these methods remain relevant and can effectively identify users. However, both Keystroke Dynamics and Browser Fingerprint methods are ineffective in cases where the False Acceptance Rate is not acceptable.

A brief summary and conclusions of the studies are presented in Table 11.

Table 11: Summary of the results

	Browser Fingerprint	Supercookie	Keystroke Dynamics
False Acceptance Rate	From 10.6% to 66.4%	0%	From 17% to 28.1%
Applicability	Need JS	Need 20+ redirects	Need JS + reason to type
Sufficient measures to prevent	Change browser	Incognito mode/clear cache/change browser	Change browser (depends on algorithm)
Frontend Performance	From 50,23 to 204,08 ms	From 2200 ms	From 1 ms

References

[1] A. Gómez-Boix, P. Laperdrix and B. Baudry, Hiding in the Crowd: an Analysis of the Effectiveness of Browser Fingerprinting at Large Scale, Proceedings of the 2018 World Wide Web Conference (2018) 309-318, https://doi.org/10.1145/3178876.3186097.

[2] P. Laperdrix, N. Bielova, B. Baudry, G. Avoine, Browser Fingerprinting: A survey, ACM Transactions on the Web (TWEB) 14(2) (2020) 1-33, https://doi.org/10.1145/3386040.

[3] Supercookie, https://supercookie.me/workwise, [28.09.2024].

[4] S. Roy, J. Pradhan, A. Kumar, D. R. Das Adhikary, A Systematic Literature Review on Latest Keystroke Dynamics Based Models, IEEE Access 10 (2022) 92192-92236, http://dx.doi.org/10.1109/ACCESS.2022.3197756.

[5] Device fingerprint. Browser fingerprint, https://en.wikipedia.org/wiki/Device_fingerprint#Browser_fingerprint, [28.09.2024].

[6] Zombie cookie, https://en.wikipedia.org/wiki/Zombie_cookie, [28.09.2024].

[7] Keystroke dynamics, https://en.wikipedia.org/wiki/Keystroke_dynamics, [28.09.2024].

[8] ISO/IEC 19795-10:2024, Information technology — Biometric performance testing and reporting, Part 10: Quantifying biometric system performance variation across demographic groups, https://www.iso.org/standard/81223.html.

[9] Navigator, https://developer.mozilla.org/en-US/docs/Web/API/Navigator, [28.09.2024].

[10] Performance, https://developer.mozilla.org/en-US/docs/Web/API/Performance, [28.09.2024].

[11] Browser-Fingerprinting via Favicon, https://supercookie.me/, [28.09.2024].

[12] KeyboardEvent, https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent, [28.09.2024].

[13] P. Eckersley, How unique is your web browser?, Proceedings of the 10th international conference on Privacy enhancing technologies (2010) 1-18.

[14] P. Laperdrix, W. Rudametkin, B. Baudry, Beauty and the Beast: Diverting modern web browsers to build unique browser fingerprints, 2016 IEEE Symposium on Security and Privacy (SP) (2016) 878-894, http://dx.doi.org/10.1109/SP.2016.57.

[15] E. Alamri, A. Alnajim, S. Alsuhibany, Investigation of Using CAPTCHA Keystroke Dynamics to Enhance the Prevention of Phishing Attacks, Future Internet 14(3) (2022) 1-21, https://doi.org/10.3390/fi14030082.

[16] A. Alsultan, K. Warwick i H. Wei, Free-text keystroke dynamics authentication for Arabic language, IET Biometrics 5(3) (2016) 164-169, http://dx.doi.org/10.1049/iet-bmt.2015.0101.