IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 17 June 2025, accepted 31 July 2025, date of publication 11 August 2025, date of current version 15 August 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3597821

RESEARCH ARTICLE

ThresholdFP: Enhanced Durability in Browser Fingerprinting

ELIF ECEM ŞAMLIOĞLU[1], SINAN EMRE TAŞÇI[2], MUHAMMED FATIH GÜLŞEN[3], AND ALBERT LEVI[1], (Senior Member, IEEE)

[1]Faculty of Engineering and Natural Sciences, Sabancı University, 34956 Istanbul, Türkiye
[2]Fraud.com International Ltd., EC3V 0HR London, U.K.
[3]İHS Kurumsal Teknoloji Hizmetleri A.Ş., 34718 Istanbul, Türkiye

Corresponding author: Albert Levi (levi@sabanciuniv.edu)

ABSTRACT Browser fingerprinting is a powerful tool for user identification in financial and other security-critical applications that require strong authentication. However, due to the instability of browser attributes, fingerprints often change rapidly, reducing their lifespan and negatively impacting user convenience. We propose ThresholdFP, a novel linking algorithm designed to extend the durability of browser fingerprints without compromising precision. Instead of replacing the fingerprint after even a small change, ThresholdFP computes a difference score between fingerprints and tolerates variations as long as they stay within a predefined threshold. To ensure realistic performance evaluation, we collected and utilized two real-world datasets. Comparative results show that the fingerprints generated by ThresholdFP are significantly more persistent than those produced by existing methods, while maintaining near-perfect precision. Notably, our approach achieves an improvement in average tracking duration ranging from 24.33% to 106.30% compared to rival schemes in the literature.

INDEX TERMS Browser fingerprinting, web authentication, tracking time, stability.

I. INTRODUCTION

Browser fingerprinting is a collection of methods used to gather various types of information from a client's browser in order to acquire a fingerprint that can be associated with the user. Browser fingerprinting methods are considered harder to detect, as they are stateless, unlike cookies, and do not store data in the client browser. Many browsers have taken measures against third-party cookies to address the controversies surrounding tracking [1]. As a result of the recent adjustments, stateless techniques such as browser fingerprinting have attracted significant popularity. Even though some browsers have also taken measures against browser fingerprinting to mitigate concerns about the potential threat it poses to user privacy, the use of browser fingerprinting by websites has been gaining momentum.

The associate editor coordinating the review of this manuscript and approving it for publication was Zijian Zhang.

However, apart from its adversarial usage, browser fingerprinting has also been used for benign purposes such as bot [2], [3], [4] and fraud detection [1]. Especially, fraud detection is a prominent goal that organizations aim to achieve with industry tools [5]. Furthermore, browser fingerprinting has started to be integrated into authentication schemes as an additional security layer and typically as a triggering mechanism to induce complementary security procedures in case of any discrepancy concerning the fingerprints [6], [7]. Such systems aim to balance user convenience and security, which elevates the importance of the uniqueness and stability of browser fingerprints. In order to be able to identify users accurately, browser fingerprinting methods need to provide unique fingerprints. Additionally, browser fingerprints are expected to be stable and long-lasting to serve their tracking purposes. On the other hand, Vastel et al. concluded that nearly half of fingerprints do not even last a week, and around 80% only last a few days

longer than a week [8]. The authors point to a series of reasons that can account for the evolution of fingerprints, including, but not limited to, software updates, relocations that cause timezone change, and some configuration changes made by the users.

Although these problems have been widely acknowledged, and amplifying the uniqueness of browser fingerprints has been the center of attention of many works in the domain [9], [10], [11], to the best of our knowledge, only a limited number of studies—most notably Eckersley's Panopticlick [12] and Vastel et al.'s FP-Stalker [8]—have investigated the means of extending the tracking time, i.e., the useful time of fingerprints, by associating them with one another.

Furthermore, given that browser fingerprinting attributes may vary in reliability, and given also the fact that some browsers have been interfering with or deprecating some of the attributes to combat browser fingerprinting, a study addressing these recent developments is needed.

To this end, in this paper, we propose ThresholdFP, a threshold-based algorithm to link fingerprints of a user to increase the tracking time. ThresholdFP calculates a difference score between the fingerprints and tolerates changes up to a threshold. To put it differently, new fingerprints with acceptable differences from previous ones are linked and are not treated as unknown. The threshold is configurable, and with the trade-off between precision and tracking time taken into account, a threshold value that is suitable for the needs of a website could be selected. Additionally, ThresholdFP carries out a remediation process if it detects that a previous decision was faulty. The algorithm ultimately aims to improve the effective lifespan of fingerprints and overcome the challenges of instability and natural evolution prevalent among fingerprints.

In addition to developing three variants of our algorithm, we adapted Vastel et al. [8] and Eckersley's work [12] to conduct a comparative analysis. Two separate datasets with different sizes, populations, and data collection frequency and intervals were employed for the analyses. All of the threshold algorithm variants outperformed the other two algorithms with the optimal thresholds, obtaining an average tracking time of 55.7 days on the first dataset and 50.1 days on the second dataset.

Furthermore, ThresholdFP is not strictly dependent on the set of fingerprinting attributes and does not necessitate regular fingerprint collection from users to facilitate linking.

The organization of the paper is as follows: Section II provides a brief literature review of the domain and elaborates on the state-of-the-art fingerprint linking studies, together with the motivation behind this study. Section III explains our approach in detail. Section IV describes the datasets used in our study and presents the results obtained. Section V discusses the results disclosed in the previous section and addresses the threats to validity. Finally, Section VI delivers the concluding remarks.

II. BACKGROUND AND MOTIVATION

The origins of browser fingerprinting date back to Mayer's 2009 study [13], in which he demonstrated that by collecting the values of a few JavaScript objects, 96.23% of users could be uniquely identified. His small-sample work was followed by Eckersley [12] in 2010, who announced the experiment on his website through various media and invited web users to participate by leaving their fingerprints. He managed to collect 470,161 fingerprints and produced the pioneer work that paved the way for browser fingerprinting to reach large masses.

Since then, numerous studies have been conducted in the field to point out the crucial role JavaScript APIs play in browser fingerprinting by allowing access to many browser-specific pieces of information such as installed fonts [14], plugins [12], [13], and screen attributes [13], [14]. Moreover, it also facilitates further identification through various APIs such as the Canvas API [15], [16], the WebGL API [10], and the Web Audio API [17] by making the browser render complex images, animations, or process audio signals to produce specific outputs.

However, as these are also common API calls that are employed for the functionality of websites, detecting the intention behind their usage may not always be clear, posing a challenge to its detection [18]. While users can disable JavaScript entirely, doing so often breaks site functionality and renders many websites unusable [19]. On the other hand, reducing browser fingerprinting to JavaScript APIs would be unfair considering that even when JavaScript is disabled, several studies reported CSS-based fingerprinting methods [20], [21], [22]. Similarly, although JavaScript does not provide an API for retrieving the browser extensions, several studies reported various workarounds for fingerprinting browser extensions [23], [24], [25].

The recent developments have escalated the privacy concerns of internet users and increased the demand for anti-tracking measures. Some of the major browsers took concrete steps in order to address these concerns. For example, Brave started to interfere with Canvas API output [26]. As a result, fingerprints became more prone to changes. This shift has, in turn, led to greater significance for fingerprinting methods that introduce mitigation strategies against these kinds of anti-tracking measures.

One of the ways of improving the durability of fingerprints is by associating evolved fingerprints with matching candidates selected from the set of previous fingerprints. In his large-scale study in 2010, Eckersley [12] formulated a basic fingerprint matching algorithm, which constituted the first attempt to link fingerprints. Each new fingerprint was compared against the set of previously encountered fingerprints. If a single candidate differed by only one attribute, the algorithm further checked whether the differing attribute belonged to either: (1) the set of attributes allowed to change regardless of value, or (2) the set of attributes permitted to change only if the string similarity ratio exceeded 85%.

The study did not measure the tracking time; however, the accuracy and the precision rates were mentioned as 65% and 99.1%, respectively.

In 2018, Vastel et al. proposed FP-Stalker that used 16 browser attributes [8] and aimed to improve on Panopticlick through a rule-based and a hybrid fingerprint matching algorithm. They imposed seven constraints in the rule-based algorithm, deeming the first three of the seven rules non-negotiable. However, the remaining four were more flexible in comparison. Nevertheless, the rules were too specifically tailored according to the observations gained from the data available. Moreover, covering all possibilities exhaustively with predetermined rules was found to be infeasible. Therefore, they employed a machine learning model in the second step of the hybrid algorithm, which is preceded by the enforcement of the first three rules of the rule-based algorithm as the first step. They also implemented Eckersley's [12] simple algorithm to compare with their approaches. The hybrid algorithm attained the best performance, while the rule-based algorithm managed to outperform Eckersley's algorithm.

On the other hand, Li et al.'s million-scale measurement study, in which they tested the performance of FP-Stalker demonstrated that the hybrid algorithm suffers from significant scalability issues [27]. Moreover, even though the rule-based algorithm was comparatively more scalable, its performance was still inadequate. Later, Pugliese et al. evaluated the effect of optimizing feature sets and removing version information on fingerprint stability [28]. They managed to double the tracking time of FP-Stalker on average. However, tailoring the feature set to maximize the tracking time rendered more than 10 browsers less trackable.

The limitations of the rule-based approaches mentioned above and the infeasibility of the learning-based approaches imply that a research direction on fingerprint durability and usability would yield a valuable contribution to the field, which we aim to address in this paper.

III. METHODOLOGY

In this section, we describe the implementation of ThresholdFP and how it is employed to achieve a longer tracking time, that is, the period for which a fingerprint is traced.

A. GENERAL WORKFLOW

In this section, the general workflow of the system will be explained. To provide a clear and comprehensive understanding of the workflow, a flowchart is shown in Figure 1, which summarizes the decision-making process in a compact manner. For a more detailed representation, Figure 2 provides a visual diagram that illustrates the interactions between the different stages of ThresholdFP.

The set of active fingerprints is the collection of a user's up-to-date fingerprints and is initially empty. Once a fingerprint is introduced, our framework first checks whether it lies in the set of active fingerprints. If so, it is marked as safe. Else, we check whether it is a previously encountered

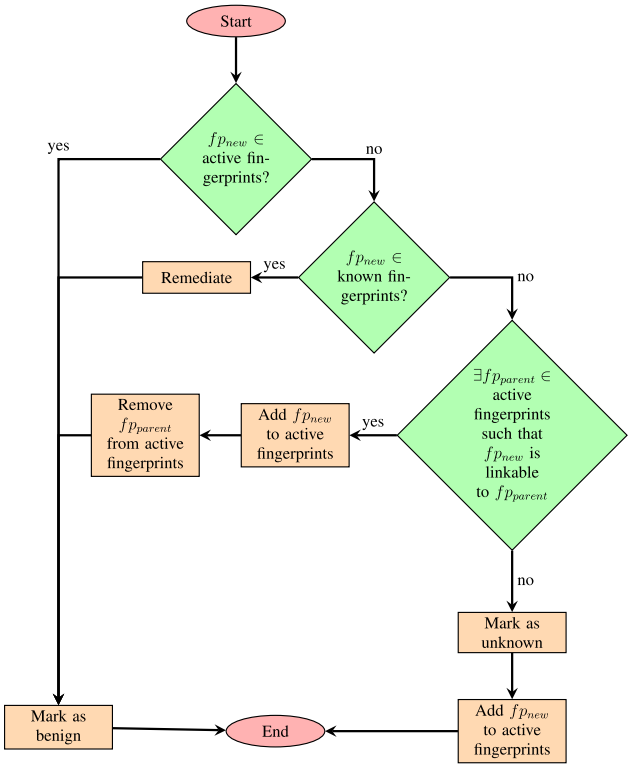


FIGURE 1. Overview of the decision-making flow in ThresholdFP.

fingerprint. If this is true, then it points out that the fingerprint was an active one before, but was later discarded from the set, meaning that until now, it was considered to be an outdated fingerprint that has been succeeded by a newer one. To follow up, a remediation process is applied to take corrective measures. The remediation process is explained in more detail in Section III-E. If the fingerprint is neither an active nor an outdated one, then it must be recorded for the first time. It can be (a) an evolved version of a previous fingerprint or (b) an unknown fingerprint from a completely new browser instance. To decide if the fingerprint can be linked to a previous fingerprint, a threshold-based algorithm is employed. If the algorithm is able to find candidates, then the candidate that shares the greatest similarity with the fingerprint is regarded as the predecessor of the fingerprint. Thereafter, the predecessor fingerprint is removed from the set of active fingerprints and the successor fingerprint is added in its place. Otherwise, if no such candidate is found, the fingerprint is treated as a new one.

B. OBTAINING THE FINGERPRINT HASH

Firstly, FingerprintJS, which is an open-source JavaScript library for fingerprinting, is loaded in the client's browser [29]. The default settings do not include the user agent header in the hash; however, the user agent header holds crucial information regarding the browser brand, version, operating system, and operating system version. Furthermore, several large-scale studies such as Panopticlick

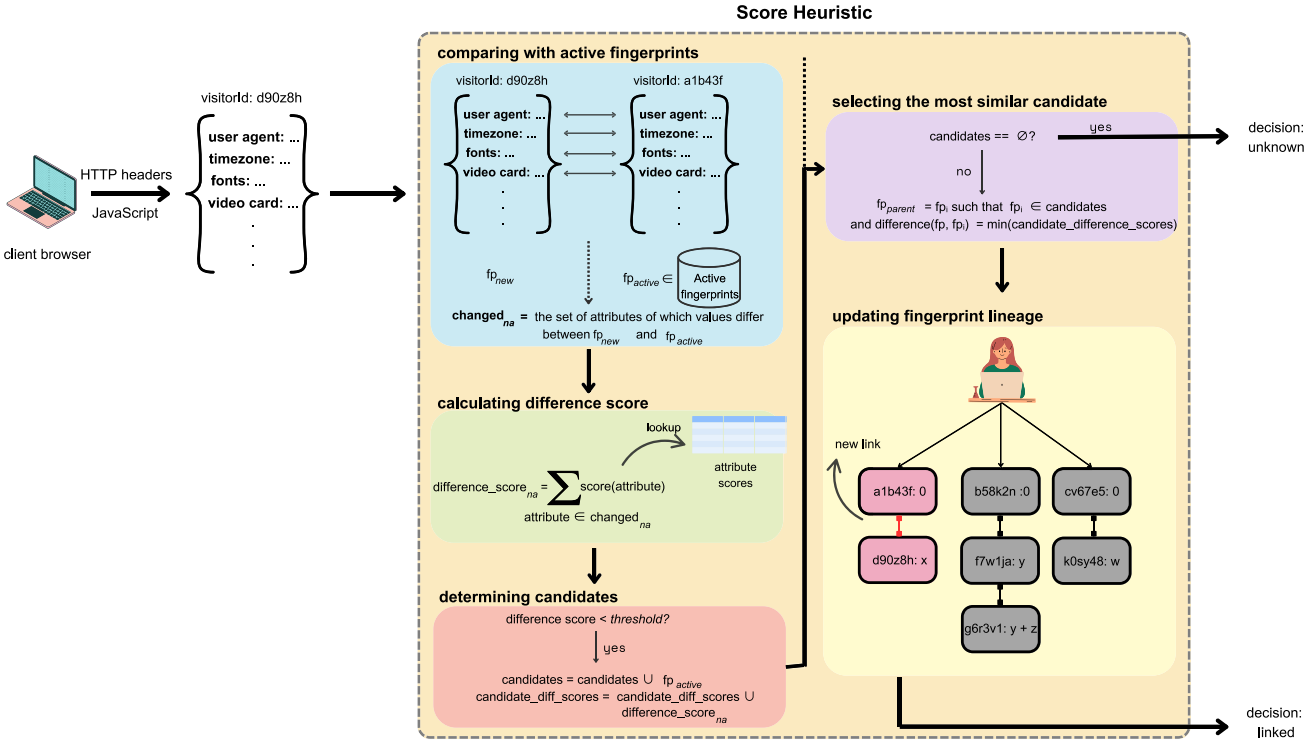


FIGURE 2. Overview of the threshold algorithm.

by Eckersley [12], AmIUnique by Laperdrix et al. [9], and Hiding in the Crowd by Gómez-Boix et al. [11] investigated the individual uniqueness of fingerprint attributes using Shannon's entropy, and suggested that high entropy values implied higher uniqueness. User agent was ranked the second, third, and second most unique attribute in these studies, respectively. In light of these factors, the user agent header was deemed valuable enough to be incorporated in the final hash in our work.

C. THRESHOLD-BASED ALGORITHM

Fingerprint hashes offer an efficient way to compare client information. However, even a single change that stems from the unstable nature of a browser attribute can result in a completely new hash. Hence, hashes provide little information on the amount and significance of the changes that have occurred. Consequently, we formed a heuristic algorithm which is based on a scoring system that tolerates the changes until the threshold is exceeded. In other words, the algorithm connects new fingerprint instances to previously observed ones in order to extend the tracking time. Algorithm 1 is provided to display a high-level implementation of the approach.

To put this heuristic algorithm into practice, each attribute is associated with a dynamic score that is inversely proportional to the number of times it evolves throughout user logins. Each individual score is intended to quantify the instability of a given attribute. The exact calculation of the attribute scores is demonstrated in Algorithm 2.

Apart from the attribute scores, we also keep the scores of fingerprints. At the time of their introduction, the score of each fingerprint is initialized to zero. If the algorithm can link the new fingerprint to a preceding fingerprint, that is, if the sum of the current score of a preceding fingerprint and the difference score between that and the new fingerprint is smaller than the threshold, a parent-child relation is formed between the previous and the new fingerprint, and the score is passed on to the child fingerprint. Later on, the child fingerprint can have its own descendant fingerprint if a new fingerprint can be linked to it. Then, the new fingerprint inherits the accumulated score and joins the lineage. The chronologically linear sequence of fingerprints that were found related to each other will be addressed as a lineage throughout this article. If there are multiple candidates that satisfy the conditions, the one which produces the minimum difference score with the new fingerprint is selected. Formally, the transmission of score to descendants could be described as the following:

$$\text{score}_{\text{child}} = \text{score}_{\text{parent}} + f(\text{child}, \text{parent}) \tag{1}$$

where f is a function that corresponds to `differenceScore` in Algorithm 1. It finds the set of attributes whose values differed between the two fingerprints and calculates the overall change score by summing the individual scores of the attributes.

D. CATEGORIZING FINGERPRINTS

The fingerprints of a user are classified into one of the three following categories: (1) appearance of an active fingerprint,

Algorithm 1 Linking Fingerprints to the Most Relevant Predecessor

```
Function differenceScore(fpAttributes, fpAttributes2):
    sum ← 0
    for attribute ∈ fpAttributes do
        val1 ← fpAttributes[attribute]
        val2 ← fpAttributes2[attribute]
        if val1 ≠ val2 then
            sum ← sum + attributeScores[attribute]
        end
    end

    return sum

Data: newFP ≠ null
minChangeScore ← 100
parentFP ← None
for activeFP ∈ ActiveFPs do
    score ← currentScore[activeFP] +
      differenceScore(newFP, activeFP)
    if score < threshold then
        if score < minChangeScore then
            minChangeScore ← score
            parentFP ← activeFP
        end
    end
end

if parentFP is not None then
    ActiveFPs ← ActiveFPs − {parentFP}
    descendantDict[parentFP] ← newFP
    currentScore[newFP] ← score
else
    currentScore[newFP] ← 0
end
ActiveFPs ← ActiveFPs + {newFP}
```

Algorithm 2 Calculating Individual Attribute Scores

```
Data: attributes ← set of all fingerprinting attributes
attributeScores ← {}
attributeChanges ← {a : 0, ∀a ∈ attributes}
totalChanges ← 0

for fpAttributes1 ∈ userFPs do
    for fpAttributes2 ∈ userFPs − {fpAttributes1} do
        anyChanges ← false
        for attribute ∈ fpAttributes1 do
            val1 ← fpAttributes1[attribute]
            val2 ← fpAttributes2[attribute]
            if val1 ≠ val2 then
                attributeChanges[attribute] ←
                  attributeChanges[attribute] + 1
                anyChanges ← true
            end
        end

        if anyChanges then
            totalChanges ← totalChanges + 1
        end
    end
end

for attribute ∈ attributes do
    attributeChange ← attributeChanges[attribute]
    percentage ← attributeChange / totalChanges × 100
    attributeScores[attribute] ← 100−percentage
end
```

(2) reappearance of an outdated fingerprint, and (3) a new fingerprint.

1) APPEARANCE OF AN ACTIVE FINGERPRINT

The set of active fingerprints is familiar to the system and are not considered a potential threat. They are marked as benign upon their arrival.

2) REAPPEARANCE OF AN OUTDATED FINGERPRINT

A fingerprint is a part of outdated fingerprints if it is among the set of fingerprints encountered so far but is not included in the set of active fingerprints. Such a case occurs if the threshold algorithm determines that fp_{new} is linked to some $fp_{previous}$ in the set of active fingerprints.

The algorithm's finding relevance between a previous fingerprint and a more recent one has two implications:

- Both fingerprints have originated from the same browser instance.
- There has been a relatively small change in the attributes of the browser instance since the previous fingerprint made its last appearance. The more recent fingerprint is the newer version of the browser instance's fingerprint and there is a predecessor-successor relation between the two fingerprints. The terms predecessor-successor and parent-child are used interchangeably throughout the paper.

In this work, we assume that each browser instance has at most one active fingerprint at any given time. However, if a fingerprint that was previously labeled as outdated reappears, it suggests that although a newer fingerprint had been marked as its successor based on similarity—leading to the original being marked as outdated—the two were not actually related. This indicates that the threshold algorithm reached a wrong conclusion about their connection. This category of fingerprints will be addressed as `mislinked` fingerprints for the remainder of the paper. Such cases may result from frequent or reversible changes, whether intentional or unintentional, that initially do not exceed the algorithm's

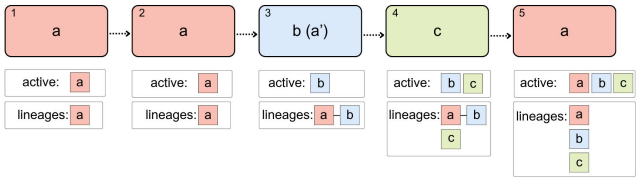


FIGURE 3. An example sequence of fingerprints from a single user, illustrating how the threshold algorithm updates the active set and lineage structure over time.

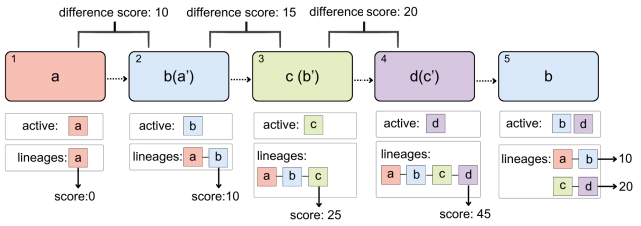


FIGURE 4. A remediation scenario.

tolerance threshold, leading it to infer a successor relationship. However, the fingerprint may occasionally revert to its former state, resulting in a reappearance. Ultimately, this category requires a remediation process, but, similar to the first category, it does not raise concerns surrounding an unfamiliar new browser instance. The remediation process will be elaborated in Section III-E.

3) NEWLY INTRODUCED FINGERPRINTS

This category includes fingerprints that have not previously been encountered. New fingerprints could result from two different reasons: a familiar browser instance producing a new fingerprint due to natural evolution, or an unfamiliar browser instance being introduced. Differentiating the former and the latter cases is not a straightforward task and demands a more complex approach than simple predetermined rules. Consequently, we employed the threshold algorithm to decide whether the new fingerprint can be associated with a previous fingerprint. In that case, the new fingerprint is marked as the successor of the previous one. These fingerprint pairs will be referred to as `linked` throughout this work. Otherwise, an unknown browser instance is believed to be used and the website is informed in case any additional security measures need to be imposed. Finally, this new fingerprint is integrated into the set of active fingerprints of the user.

An illustrative example of how these fingerprint categories emerge in practice is provided in Figure 3. The figure demonstrates a sequence of fingerprints collected from a single user over five discrete time steps, and how each is classified into one of the three defined categories. It also shows how these classifications affect the set of active fingerprints and the lineage structure maintained by the threshold algorithm.

In both Figures 3 and 4, each box represents a fingerprint collected at time step t (indicated in the top-left corner), with the fingerprint hash shown inside. Below each box, *active* indicates the set of active fingerprints after processing the fingerprint, and *lineages* shows the updated lineage structure.

- $t = 1$**:** Fingerprint "`a`" is observed for the first time. Since there are no previous fingerprints, it is classified as a **new fingerprint** (Category 3). It is added to the active set, and a new root node is created for it in the lineage structure.
- $t = 2$**:** Fingerprint "`a`" appears again. This fingerprint exactly matches the current active fingerprint from the previous step. It is therefore classified as an **appearance of an active fingerprint** (Category 1). No updates are made to the active set or the lineage.
- $t = 3$**:** A **new fingerprint** (Category 3) "`b`" arrives. The algorithm determines that "`b`" is sufficiently similar to the active fingerprint "`a`", and considers it a possible update. As a result, the active set is updated by replacing "`a`" with "`b`", and the lineage structure is updated by linking "`b`" as a successor of "`a`".
- $t = 4$**:** Another **new fingerprint** (Category 3), "`c`", is received. However, it is not similar enough to any existing fingerprint in the active set. Since it cannot be linked to a known fingerprint, it is marked as *unknown*, and this status is returned to the caller. Then, it is added to the active set, and a new root node is created for it in the lineage structure.
- $t = 5$**:** Fingerprint "`a`" reappears. At this point, "`a`" was previously removed from the active set after being linked to "`b`" as its assumed successor. However, the reappearance of "`a`" indicates that this assumption was incorrect. If "`b`" had truly been a successor of "`a`", then no further instances of "`a`" would be expected. Therefore, "`a`" is classified as a **reappearance of an outdated fingerprint** (Category 2). The algorithm adds "`a`" back to the active set and removes the previously established parent-child link between "`a`" and "`b`" in the lineage structure, treating them as independent root fingerprints.

E. REMEDIATION PROCESS

As mentioned in Section III-D2, the comeback of a fingerprint that was marked as outdated necessitates some corrections. Such a fingerprint can have:

no ancestors but descendants
both ancestors and descendants

No case without descendants is possible, because then the reappeared fingerprint would not have been labeled as outdated in the first place.

The treatment of both cases is identical. Firstly, the parent-child relationship between the reappeared fingerprint and the descendant one is broken. Then, the current score of the descendant, which is the sum of the scores of the changed attributes between the reappeared fingerprint and the descendant, is reverted back to 0. If the descendant has descendants itself, then throughout the whole lineage, the same amount is subtracted from the scores of remaining descendants to negate the effect of the parent fingerprint.

The reason the same resolution applies to the second case is that the relationship between the ancestor and the reappeared fingerprint is not a part of the erroneous decision. Consequently, ancestor fingerprints should preserve the link to the reappeared fingerprint.

Figure 4 is provided to depict a scenario in which the reappeared fingerprint has both an ancestor and a sequence of descendants. At $t = 2$, Fingerprint "b" is linked to Fingerprint "a" and is assigned the difference score between them, which is calculated as 10. After that, Fingerprint "c" arrives. Likewise, it is linked to Fingerprint "b". It inherits the score of Fingerprint "b", and a difference score of 15 is added on top of that, accumulating Fingerprint "c"'s score to 25 as demonstrated in the figure. This score is passed on to Fingerprint "d" after it gets linked to Fingerprint "c", with an addition of 20 in regards to their difference score. At the end, Fingerprint "b" is observed to make a reappearance. This does not disrupt its logical relevance to Fingerprint "a"; therefore, its score and link to Fingerprint "a" are preserved. On the other hand, although the bond between Fingerprint "c" and Fingerprint "d" remains intact, they are revealed to be unrelated to Fingerprint "b" by this recent development. Thereafter, the link between Fingerprint "b" and Fingerprint "c" is broken, creating another lineage with Fingerprint "c" as the root and Fingerprint "d" as its child. Finally, the score of Fingerprint "c" is subtracted from the score of Fingerprint "d", and then Fingerprint "c"'s score is reset to 0 in order to cancel the contribution of the erroneous link.

IV. DATASETS & RESULTS

In this section, a comprehensive view of the datasets is offered, and the results of the study are delivered. In this context, "linking" refers to the act of associating a newly received fingerprint with a previously seen one, forming a continuity chain or lineage of fingerprints that are presumed to belong to the same browser instance.

A. EVALUATION METRICS

In this section, the evaluation metrics used to assess the performance of our work are defined.

- **True Positive (TP):** Number of `linked` fingerprint pairs where the parent and child have the same browser label, in other words, number of accurate links established between fingerprints.
- **Estimated True Positive (ETP):** Number of `linked` fingerprints minus number of `mislinked` fingerprints.
- **False Positive (FP):** Number of `linked` fingerprint pairs where the parent and child have different browser labels, in other words, number of pairs between which the threshold algorithm formed an inaccurate link.
- **Precision:** The ratio of the number of true positives (TP) to that of all positive predictions, i.e., the sum of $TP+FP$.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

- **Estimated Precision:** The ratio of the number of estimated true positives (ETP) to that of all positive predictions, i.e., the sum of $TP + FP$.

$$\text{Estimated Precision} = \frac{\text{ETP}}{\text{TP} + \text{FP}} \tag{3}$$

- **Tracking Time:** Time passed between the first encounter of the root ancestor and the last encounter of the latest descendant in a fingerprint lineage. To put it differently, the amount of time that a fingerprint, specifically the root ancestor in a lineage, can be identified.
- **Matching Time:** Time to process and return a decision for a single fingerprint, that is, a decision on whether it can be linked to a previous one or not.

Our hypothesis is that as the threshold increases, the algorithm tolerates more changes, which in turn extends the tracking time. However, this increased tolerance might lead to the establishment of inaccurate links between fingerprints. As a result, both precision and estimated precision are expected to deteriorate. This trade-off highlights the importance of optimizing the threshold value to achieve a balance between the metrics. This, in fact, is what we analyze in the rest of this section.

B. DATASETS

The same set of fingerprint attributes was collected for both of the datasets. Tables 6 and 7 in the Appendix provide an explanatory overview of the fingerprint attributes used. The descriptions and examples provided for the attributes in the table are prepared in accordance with the source code of FingerprintJS library and the "Web APIs" page of Mozilla Developer Network [30], and may correspond to different information in different resources.

Moreover, individual scores of attributes that are used in the threshold algorithm, as well as the normalized Shannon entropy for each of them, were computed separately for each dataset and are listed in Table 1 and Table 2.

The individual attribute scores are calculated based on how frequently each attribute changes across a user's fingerprint visits. Attributes that change less frequently are assigned higher scores, reflecting their stability and their contribution to fingerprint durability.

The Shannon entropy values, in contrast, are included for descriptive purposes only and are not used in the scoring process. Entropy is computed from the distribution of attribute values across all users in the dataset and reflects attribute uniqueness. A loose inverse relationship between the attribute scores and the Shannon entropy values has been observed. This is an expected trend, though not guaranteed, as the two metrics capture different aspects: temporal stability for attribute scores and global uniqueness for entropy.

1) INSTITUTIONAL DATASET

This dataset was collected through the website of an ICT (Information and Communications Technology) company

TABLE 1. Normalized Shannon entropy and calculated scores of attributes in the institutional dataset.

Attribute	Score	Shannon entropy (normalized)
architecture	92.48	0.49
audio	66.29	0.18
canvas	20.08	0.65
colorDepth	93.42	0.21
colorGamut	82.40	0.30
contrast	97.05	0.03
cookiesEnabled	0	0.00
cpuClass	0	0.00
deviceMemory	70.05	0.48
domBlockers	95.47	0.08
fontPreferences	68.96	0.22
fonts	40.95	0.64
forcedColors	98.69	0.08
hardwareConcurrency	57.33	0.48
hdr	89.35	0.31
indexedDB	99.99	0.00
invertedColors	96.39	0.29
languages	66.35	0.29
localStorage	0	0.00
math	78.63	0.20
monochrome	0	0.00
openDatabase	80.61	0.52
osCpu	81.41	0.10
pdfViewerEnabled	95.08	0.14
platform	76.62	0.24
plugins	82.77	0.10
reducedMotion	92.91	0.24
screenFrame	44.92	0.35
screenResolution	49.51	0.36
sessionStorage	0	0.00
timezone	94.63	0.10
touchSupport	93.82	0.10
userAgent	9.60	0.51
vendor	79.80	0.40
vendorFlavors	79.51	0.23
videoCard	39.84	0.65

based in Türkiye. The dataset consists of 271,000 fingerprints of 32,052 users that were collected over a period of 5.5 months between May 2023 and October 2023. 35,578 out of 271,000 fingerprints that were retrieved from mobile devices were excluded from the study, leaving 28,467 users. Every fingerprint instance in the dataset has a unique user identifier of the account owner; however, the dataset does not include browser identifiers such as cookies. Although many browser fingerprinting studies in the domain rely on cookies as ground truth [9], [11], [12], they were recently shown to be unreliable for use as identifiers [27]. Similarly, Pugliese et al. emphasized the inadequacy of cookies for constituting reliable identifiers and employed a user-level identifier instead [28]. Nevertheless, due to the lack of browser identifiers, the performance of the threshold algorithm on the institutional dataset was evaluated via an estimation metric, which will be further elaborated in Section IV-C1.

Table 1 features the scores and normalized Shannon entropy values calculated for attributes with respect to this dataset.

TABLE 2. Normalized Shannon entropy and calculated scores of attributes in the volunteer dataset.

Attribute	Score	Shannon entropy (normalized)
architecture	70.42	0.87
audio	48.37	0.53
canvas	19.99	0.82
colorDepth	84.09	0.52
colorGamut	68.11	0.94
contrast	94.96	0.15
cookiesEnabled	0	0
cpuClass	0	0
deviceMemory	69.81	0.51
domBlockers	80.14	0.53
fontPreferences	58.77	0.55
fonts	44.91	0.68
forcedColors	90.57	0.31
hardwareConcurrency	51.75	0.71
hdr	67.63	0.58
indexedDB	0	0
invertedColors	80.14	0.79
languages	58.23	0.75
localStorage	0	0
math	71.77	0.55
monochrome	0	0
openDatabase	97.56	0.11
osCpu	95.93	0.28
pdfViewerEnabled	89.64	0.32
platform	64.388	0.69
plugins	80.11	0.21
reducedMotion	91.63	0.45
screenFrame	31.12	0.67
screenResolution	35.26	0.71
sessionStorage	0	0
timezone	89.48	0.15
touchSupport	92.85	0.20
userAgent	15.56	0.82
vendor	77.80	0.67
vendorFlavors	77.80	0.67
videoCard	36.45	0.82

2) VOLUNTEER DATASET

This labeled dataset of 5,774 fingerprints was collected from volunteers at an academic institution, most of whom were undergraduate students. The participants were instructed to visit a fingerprinting website that collects the same fingerprinting attributes as those present in the former dataset on a daily basis from two different devices between March 7 and May 19, 2024. Unlike the former dataset, each fingerprint was labeled with the name of the device by the participants themselves, in efforts to compile a dataset with which not the estimated but the exact performance of the threshold algorithm could be measured. The dataset was filtered to exclude 70 instances where a different browser was utilized on the same device, to ensure that the labels identify single browser instances and establish a reliable ground truth, with each device label also serving as a browser label/identifier.

The individual scores and normalized Shannon entropy values computed for the fingerprint attributes based on this dataset are presented in Table 2.

TABLE 3. Distribution of operating systems across two datasets.

Operating System	Institutional (%)	Volunteer (%)
Windows	79.0	64.6
MacOS	14.9	34.2
Linux	6.1	1.2

TABLE 4. Distribution of browsers across two datasets.

Browser	Institutional (%)	Volunteer (%)
Chrome	68.6	61.3
Firefox	14.2	3.4
Safari	5.1	21.1
Edge	7.6	5.1
Opera	3.8	9.1
Other	0.7	-

C. RESULTS ON THE INSTITUTIONAL DATASET

This section presents the performance of ThresholdFP on the institutional dataset, measured by precision and tracking time.

1) PRECISION

As mentioned in Section III-D2, the threshold algorithm may occasionally establish incorrect bonds between fingerprints. However, since the fingerprints were received without any label or attribute that could act as ground truth, another way of measurement was needed to unravel the errors of the threshold algorithm. Hence, the number of reappeared outdated fingerprints, which account for errors that had observable consequences, was measured as an estimation of the number of errors.

In order to test how ancestor selection in cases of multiple candidates performs, three variants of the threshold algorithm were implemented. The `earliest` variant selects the fingerprint that comes first in chronological order, the `min` variant selects the fingerprint that has the minimum difference score with the successor, and lastly, the `max` variant selects the fingerprint that has the maximum difference score with the successor. Figure 5 displays the precisions of all three variants of the threshold algorithm.

The reason behind the similar results has been observed to be that there is a single parental candidate in most of the cases. Therefore, the method of ancestor selection does not matter significantly. Nevertheless, we proceeded with the `min` variant for the comparative analysis, as it was both the most intuitive and the best-performing variant.

2) TRACKING TIME OF FINGERPRINTS

The tracking time is calculated for fingerprint lineages consisting of at least two fingerprints. The root fingerprint's first recorded arrival time and the leaf fingerprint's last recorded arrival time constitute the boundaries for the calculation, and the duration between them is measured as the tracking time of the root fingerprint.

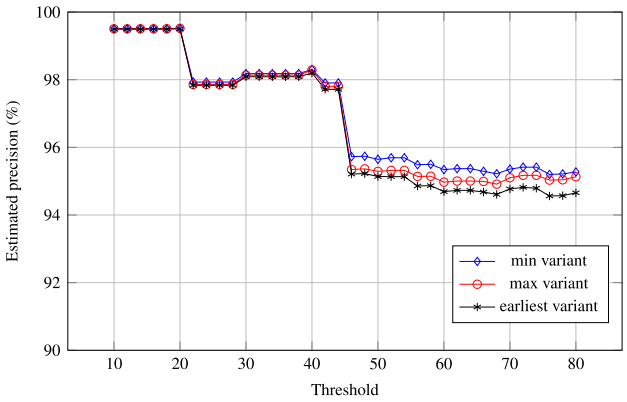


FIGURE 5. Estimated precision of three threshold algorithm variants on the institutional dataset.

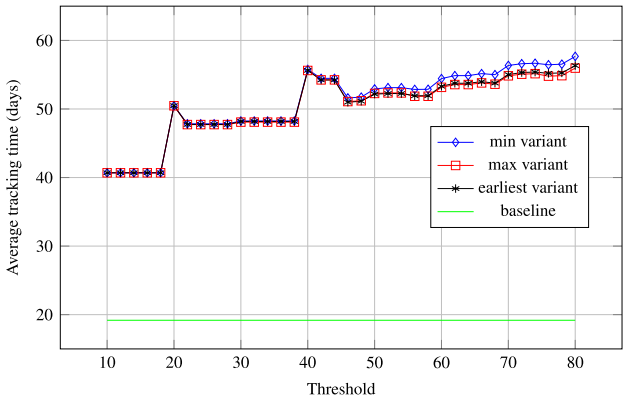


FIGURE 6. Average tracking time of three threshold algorithm variants on the institutional dataset.

In order to compare the results of our work to a baseline, the average duration of the fingerprints without any algorithmic intervention was calculated. For every fingerprint, its initial appearance marked the kickoff time. After that, the time until a completely new fingerprint that was not encountered before arrived was measured. The time calculated was regarded as the original duration of the fingerprint. The process was repeated for every fingerprint, and the average time was obtained as 19.1 days. Here, we assumed that all of the previously seen fingerprints are remembered and not treated as new, allowing for the maximization of the measurements, which yields the best-case performance of the baseline.

Figure 6 displays the performance of all three variants of the threshold algorithm across different thresholds. The baseline duration is a constant value that is not dependent on the threshold.

In line with the precision results, the average duration of the fingerprints did not remarkably differ between the algorithm variants. The curve followed a generally increasing trend, and the best performance was obtained by the `min`

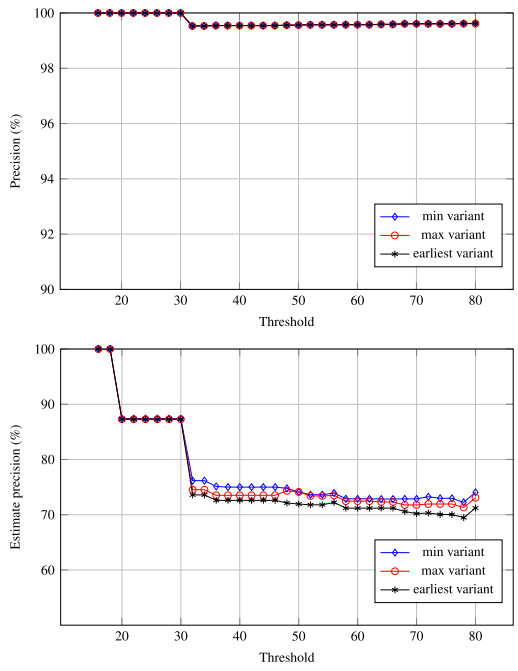


FIGURE 7. Precision and estimate precision of three threshold algorithm variants on the volunteer dataset.

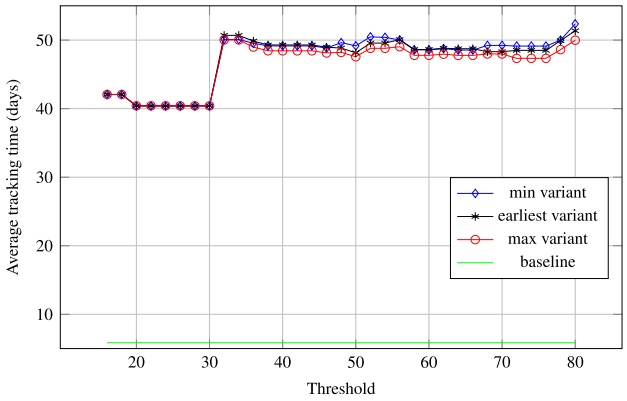


FIGURE 8. Average tracking time of three threshold algorithm variants on the volunteer dataset.

variant. An average tracking time of 40.6, 55.7 and 57.6 days was recorded when threshold was set to 10, 40 and 80, respectively. All three versions managed to extend the tracking time to more than two to three times the original duration of 19.1 days, depending on the threshold.

D. RESULTS ON THE VOLUNTEER (LABELED) DATASET

This section presents the performance of ThresholdFP on the volunteer dataset in terms of precision and tracking time. Unlike the figures for the previous dataset, the figures for the volunteer dataset start the threshold from 16 instead of 10. Since the smallest attribute score is 15.56, the difference scores calculated between fingerprints always exceed the limit when the threshold is set to values lower than this score, and consequently, no link gets established. Therefore, precision and tracking time were measured only for threshold values larger than 15.

1) PRECISION

Besides the precision estimation in Section IV-C1, an exact measurement of the ThresholdFP's precision on this dataset was made possible by the availability of browser labels.

As evident in Figure 7, the actual precision and the estimated precision differed significantly: the actual precision never dropped below 99%, whereas the estimated precision experienced sharp declines and started to stabilize around 70%. Measurement of the estimated precision on the institutional dataset did not produce such an outcome, however. It remained above 94% even at high thresholds as depicted in Figure 5. A possible explanation for such dissimilar observations in the results of two datasets may lie in the population of the datasets, more specifically, the distribution of operating systems and browser brands, featured in Table 3 and Table 4, respectively. Considering that estimated precision decreases as the number of `mislinked` fingerprints increase, some operating systems or browser brands with built-in privacy features (e.g., Safari), which are more prevalent in this dataset, might be interfering with the fingerprinting results, leading to an escalated number of `mislinked` fingerprints. We leave this as future work.

2) TRACKING TIME OF FINGERPRINTS

The tracking time of ThresholdFP on the volunteer dataset was measured with the same approach discussed in Section IV-C2.

Figure 8 displays that the `min` variant and the `earliest` variant alternately took the leading position, unlike the corresponding results for the first dataset where the `min` variant consistently maintained the top position. At the smallest and the largest thresholds, the average tracking time attained with the `min` variant was 42.1 and 52.4 days. In between, it was observed to peak around threshold values 32 and 52, corresponding to 50.1 and 50.5 days of average tracking time, respectively.

E. COMPARATIVE ANALYSIS

For our comparative analysis, we picked two major studies, namely FP-Stalker by Vastel et al. [8] and Panopticlick by Eckersley [12]. This section breaks down the process of applying their methods to our datasets and provides a comparison of the results. The common contributive characteristic of these two studies is that they investigated means of linking fingerprints to extend the tracking duration. Although there are some other studies on linkability such as [27] and [28], and some industrial tools [5] for fingerprinting, they either discuss some optimizations and empirical results, rather than proposing algorithmic novelty, or are not replicable using existing datasets.

Eckersley proposed a simple algorithm in 2010 which examines if a previous fingerprint may have evolved into the newly arrived fingerprint. He makes use of 8 attributes, namely: `user agent`, `accept`, `cookies enabled`, `screen resolution`, `timezone`, `plugins`, `fonts`, and `local storage`. The algorithm compares the previous fingerprints with the new fingerprint to check which attribute values have changed. If only one attribute is changed, the previous fingerprint is considered a candidate. After all of the previous fingerprints are considered, if there is a single candidate, the new fingerprint is linked to it if one of the following holds: (1) the changed attribute belongs to the set {`cookies enabled`, `screen resolution`, `timezone`, `local storage`} or (2) the attribute values bear a similarity value above 0.85.

In 2018, Vastel et al. proposed FP-Stalker, introducing one completely rule-based algorithm and one hybrid algorithm, which combines the rule-based one with machine learning methods to match new fingerprints with already encountered ones. The rule-based algorithm applies a predetermined set of seven rules, whereas the hybrid algorithm first applies the first three rules of the rule-based algorithm and then employs a random forest model. The hybrid algorithm was observed to perform better but was found to be unscalable [27]; therefore, we proceeded with the rule-based algorithm in the comparative analysis.

There were a number of fundamental differences between the previous studies and our approach. Firstly, while a few attributes, namely `DoNotTrack`, `accept` and `encoding` are not available within our datasets, they include significantly more attributes than FP-Stalker and Panopticlick. Thus, not only do the studies not share the same set of attributes, but also the algorithms of Vastel et al. and Eckersley were heavily dependent on the collected attributes. For instance, a subset of attributes is allowed to change, whereas another subset of attributes is permitted to change only if the change ratio is below a certain threshold, and for some attributes, the algorithm mandates no changes. Thus, assigning the extra attributes collected in our study to one of such subsets arbitrarily, in an effort to make a complete adaptation would have introduced subjectivity and risked inconsistency. As a result, in order to implement both algorithms accurately, the dataset was cleared of the extra attributes, leaving only the set of attributes present in each of the studies. Both of the algorithms were adapted from the GitHub repository referenced in the FP-Stalker study.[1]

Secondly, Panopticlick leveraged HTTP cookies for browser identification that were persistent up to three months. In FP-Stalker, the users had installed browser extensions that uniquely identified the browser instance. As mentioned earlier, the institutional dataset includes a user identifier number, but unlike the volunteer dataset, it does not contain a device or browser identifier. As the estimated precision metric proposed earlier is specific to the threshold algorithm, the precision measurement of previous work was only applicable to the volunteer dataset. Additionally, under some conditions, the rule-based algorithm of FP-Stalker imposes an additional homogeneity check for the browser identifiers of the candidates, which was suitable for the volunteer dataset but not for the institutional dataset. Therefore, to be able to compare the performances of all algorithms on the institutional data, two variants of the rule-based algorithm were evaluated, one yielding an upper bound and the other yielding a lower bound on the tracking time. The first variant is configured to always act as if the candidates belong to the same browser. This results in a maximum tracking duration. Conversely, the second variant proceeds as if the homogeneity check always fails, which keeps the number of links and the tracking time at a minimum.

Finally, Eckersley and Vastel et al. compare the new fingerprint with the whole set of fingerprints received so far, whereas, due to our use case, we only compare the new fingerprint to the previous fingerprints of the corresponding user. Thus, the algorithms proposed by Eckersley and Vastel et al. were modified to incorporate this consideration.

1) AVERAGE TRACKING TIME

Table 5 summarizes the average tracking time of the algorithms on both datasets. For the comparison, ThresholdFP's threshold value was set to 40 and 32 for the institutional dataset and the volunteer dataset, respectively. These threshold values were determined independently for each dataset based on the empirical testing presented in the Datasets & Results section, balancing the trade-off between precision and tracking time to achieve optimal performance. The best performance for the average tracking time on the institutional dataset was achieved by ThresholdFP with 55.7 days and over 98% estimated precision. Since the estimated precision was specifically designed to account for the mistakes of the threshold algorithm, it was not applicable to the other linking algorithms. Panopticlick surpassed the lower bound of FP-Stalker by 7.5 days, but FP-Stalker's upper bound managed to outperform Panopticlick by 10.3 days. Regarding the analysis with the volunteer dataset, the algorithms were observed to perform better than on the institutional dataset, relative to the span of the data collection. This might be rooted in the frequency and the regularity of the data collection: fingerprints in the volunteer dataset were collected almost every day. Similar to the results for the institutional dataset, ThresholdFP outperformed the other two studies with 50.1 days of average tracking time. In the volunteer dataset, as the ground truth was available, FP-Stalker could be adapted faithfully, allowing for the homogeneity check mentioned above and therefore, rendering the exact performance measurement possible. In contrast to the findings of Vastel et al. [8], Panopticlick displayed better performance than FP-Stalker, exceeding it by 6.8 days. On the other hand, the findings demonstrated that FP-Stalker achieved a precision of 97.6%, whereas Panopticlick achieved a precision of 92.3%. Surpassing

[1] Open source at: https://github.com/Spirals-Team/FPStalker

TABLE 5. Comparison of the average tracking time obtained in number of days by ThresholdFP and previous work.

Dataset	ThresholdFP	Panopticlick [12]	FP-Stalker [8]	
			lower bound	upper bound
Institutional	55.7	34.5	27	44.8
Volunteer	50.1	45.6	38.8	

both, ThresholdFP obtained an almost perfect precision of 99.5%

Across both datasets and all three prior methods (Panopticlick, FP-Stalker-upper, FP-Stalker-lower), ThresholdFP consistently achieved longer tracking durations. The relative increase ranged from 24.33% (compared to the FP-Stalker upper bound) to 106.30% (compared to the FP-Stalker lower bound) on the institutional dataset, with Panopticlick comparisons and all corresponding results from the volunteer dataset also falling within this range. We report this as a representative performance range in the Abstract.

2) SCALABILITY

Li et al. [27] demonstrated that FP-Stalker is not scalable in a large-scale setting. Their experiments show that with as few as 200,000 fingerprints, the rule-based algorithm struggles to meet the latency expectations of real-time bidding (RTB), where ad decisions are typically required in under 100 milliseconds [31].

In our setting, as the user identifiers are provided together with the fingerprints, a new fingerprint is only compared to the user's previous fingerprints. Therefore, we evaluated scalability based on the number of users instead of the number of fingerprints, since the latter did not consistently impact algorithm runtime in this setting. Matching time was measured in a realistic environment by adapting both algorithms to include database connections and associated overhead.

The experiments were run on an AWS EC2 m6i.2xlarge instance with 32 GB DDR4 memory and an Intel® Xeon® Platinum 8375C (Ice Lake) processor with up to 3.5GHz turbo boost frequency.

We used the lower-bound variant of FP-Stalker's rule-based algorithm, although this choice does not affect the matching time measurements reported in our scalability evaluation.

The average matching time remained consistently low for both algorithms, with values generally below 1 ms and only occasionally exceeding it, significantly under the 100 ms threshold required by RTB systems. To enable comparison with Li et al.'s findings [27], we note that at the 28k user mark on the x-axis, over 230,000 data points were processed in total for calculating the "average matching time" on the y-axis.

While our approach introduces only a marginal increase in latency compared to FP-Stalker, it delivers a significant improvement in performance, extending the average tracking time by at least 24.3%. This notable gain highlights the effectiveness of our algorithm, especially given that the average matching time remains consistently within just 0.3-0.4 ms of FP-Stalker. The slight latency overhead is thus a minimal cost for a substantial enhancement in tracking longevity, making our method a compelling alternative.

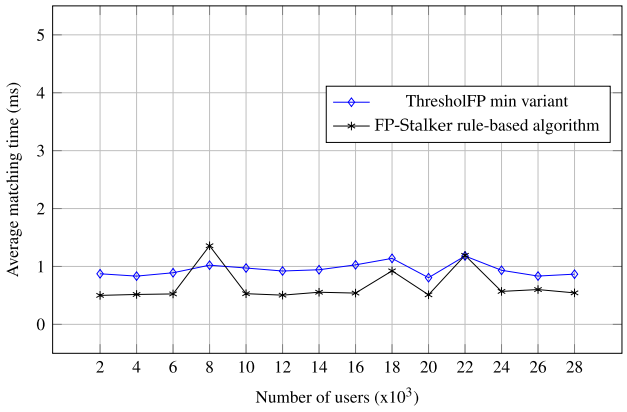


FIGURE 9. Average decision time of ThresholdFP and FP-Stalker's rule-based algoritm for one fingerprint.

V. DISCUSSION & THREATS TO VALIDITY

Although it is intuitive to expect a monotonically increasing trend in average tracking time as the threshold increases, our models occasionally demonstrated fluctuating behavior. We identified several cases that contributed to such an outcome. For instance, larger thresholds bear an elevated chance of one fingerprint getting associated with another one that would not normally be considered similar. Consequently, as the difference scores that the threshold algorithm allows grow larger, the threshold merit of the lineage might get used up sooner, resulting in a shorter tracking time. Similarly, since the algorithm becomes more tolerant, the number of inaccurate links, which later get broken in remediation processes, increases and it gives rise to shorter fingerprint lineages. The combined effect of these cases may explain the formation of local minima in the curves.

An important observation was that, even though the volunteer dataset was gathered over a shorter period of time, the measurements of average tracking time on this dataset turned out to be close to, or even better than, those conducted on the institutional dataset for all of the algorithms. This outcome is in support of the finding of Vastel et al. that higher collection frequency results in a higher average tracking time for the rule-based algorithm and Eckersley's algorithm.

Analyses across different datasets also show that individual attribute scores and optimal threshold values may vary. Thus, ThresholdFP is also dependent on data, just as the rival mechanisms it was compared to. However, this dependency is not continuous. When integrating ThresholdFP into a live system, default scores and a provisional threshold value can be used initially. As the system runs, the collected data can be used for fine-tuning as an offline process, and then the

parameters can be updated. Alternatively, the system can first collect data to determine attribute scores and a threshold, and then initialize the algorithm using these values.

The expected trade-off between precision and tracking time is evident in the results. In other words, while tracking time increases, precision decreases with threshold. The non-monotonic behavior of both tracking time and precision metrics may even be helpful to choose optimal threshold values based on the system administrator's preference by picking threshold values of the right edges of the target metric value.

In order to address possible limitations, we considered threats to internal, external, and construct validity:

Threats to internal validity: For the reasons discussed in Section IV-E, the algorithms of Eckersley [12] and Vastel et al. [8], in their original shape, were not suitable for our analyses, rendering our intervention inevitable. Despite efforts to faithfully replicate the original algorithms, it is possible that unintentional errors were introduced during adaptation.

Furthermore, the GitHub repository provided in Vastel et al.'s work was used as the source material. If there was a bug in the source code, it might have propagated to our adaptations.

Threats to external validity: We analyzed the performance of all algorithms on two separate datasets that were collected independently. The institutional dataset was gathered by an ICT company, and the volunteer dataset had all of its participants from an academic environment. In addition to the population, the frequency of the data differed remarkably, reducing the chance of bias. The volunteer dataset consisted of fingerprints collected daily, with occasional exceptions, unlike the institutional dataset, which was composed of fingerprints collected at arbitrary times. Regardless of such differences, our approach performed the best in both of the comparative analyses employing either one of the datasets, and the performances of other algorithms were consistent across both datasets. Nevertheless, the participants in our study may not represent the broader user population, and our approach might not yield the same performance with different datasets.

Additionally, the time span of data collection was 2.5 months and 5.5 months, which might be deemed short in comparison to the prominent studies in the field. Finally, due to privacy reasons, we are unable to share the datasets, and that prevents the reproducibility of this work. Nonetheless, this is a common limitation in fingerprinting studies.

Threats to construct validity: Since the fingerprints in the institutional dataset did not contain browser identifiers, we formulated an "estimated precision" metric, as explained in Section IV-A and Section IV-C1. There is a possibility that this metric might not model the actual precision successfully or that it might produce inflated results. However, the analysis on the volunteer dataset revealed that actual precision was significantly better than the estimate precision, as illustrated in Figure 7, indicating that the estimated precision was not an overestimate of the actual precision.

VI. CONCLUSION

In this paper, we proposed ThresholdFP, a threshold-based fingerprint linking algorithm, that associates new fingerprints with previously encountered ones. Fingerprints can change due to factors such as attribute instability, natural evolution, and user intervention. ThresholdFP aimed to mitigate these challenges in order to achieve an extended tracking time for the fingerprints.

To evaluate ThresholdFP's performance, two datasets were generated and used. The first dataset was composed of 235,422 fingerprints collected over a span of 5.5 months. The second data collection phase, lasting a total of 2.5 months, was initiated to form a secondary dataset. Unlike the first dataset, this one includes manually labeled browser identifiers to be utilized as ground truth. In the end, the second dataset contained 5,774 fingerprints in total.

To decide which strategy to follow when multiple candidates are present to associate with a fingerprint, three variants of our threshold-based algorithm were created. Furthermore, all three variants were compared against two previous studies [8], [12], which also put forward rule-based algorithms to link fingerprints.

A separate analysis was conducted for each of the datasets. In the analysis of the first dataset, instead of an exact measurement, upper and lower bounds were calculated for FP-Stalker. This is due to the fact that the original algorithm applied a check that was not applicable to our dataset. Therefore, we mimicked its behavior to maximize and minimize the average tracking time, obtaining the upper and lower bounds, respectively. The ThresholdFP variant that selects the candidate with the least difference to the new fingerprint yielded the best performance when the threshold was set to 40. This threshold value achieved an average tracking time of more than 55 days, with an estimated precision of over 98%, compared to 34.5 days by Panopticlick and 27 to 44.8 days by FP-Stalker.

The same variant of ThresholdFP also achieved the best performance on the second dataset, with 50.1 days and 99.5% precision, surpassing Panopticlick and FP-Stalker, which attained an average tracking time of 45.6 and 38.8 days, and a precision of 92.3% and 97.6%, respectively.

VII. ETHICS CONSIDERATIONS

The attributes were obtained via FingerprintJS, which is a well-known open-source browser fingerprinting library that is widely utilized by websites all around the world [25]. Apart from the attributes retrieved by FingerprintJS, no personal information was gathered from the users. Hence, none of the datasets contain any private information.

The attributes were already being collected for a browser fingerprinting product employed by the ICT company due to security reasons. Data collection for our experiment was

TABLE 6. Fingerprint attributes (Part 1).

Attribute	Source	Description	Example
User agent	HTTP header	User agent string identifying browser and OS	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
Fonts	JavaScript	List of supported system fonts	Agency FB, Calibri...
Font preferences	JavaScript	Measured text widths using sample fonts	"default": 149.3125...
Audio	JavaScript	Sum of absolute sample values from a rendered audio buffer generated with Web Audio API	124.04
Screen resolution	JavaScript	Width and height of the screen	[720, 1280]
Screen frame	JavaScript	Dimensions of the visible browser window frame	[40, 0, 70, 0]
Color depth	JavaScript	Number of bits used per pixel for color	24
Timezone	JavaScript	Local time zone setting	Europe/Istanbul
Session storage	JavaScript	Availability of session-based storage	true
Local storage	JavaScript	Availability of persistent local storage	true
DOM blockers	JavaScript	Detected content blockers	adGuardBase
Platform	JavaScript	Platform identifier string	Win32
OS CPU	JavaScript	Reported operating system and version	Windows NT 10.0
Architecture	JavaScript	CPU architecture indicator	255
CPU class	JavaScript	CPU classification (legacy attribute)	x86
Hardware concurrency	JavaScript	Number of logical CPU cores	8
Device memory	JavaScript	Available device memory for browser	8
Indexed DB	JavaScript	Support for IndexedDB API	true
Open database	JavaScript	Support for Web SQL database	true

conducted under the existing consent agreements and setup. With regard to the volunteer dataset, the participants became a part of the experiment voluntarily after they were informed about the data collection and their consent was received. Moreover, our system was designed in accordance with data minimization principles, and the fingerprint attributes collected are unrelated to the services provided on the websites, thereby not facilitating any behavioral or personal profiling.

VIII. ETHICAL IMPLICATIONS

This work aims to improve the persistence of browser fingerprints to support legitimate security use cases, such as fraud detection, device verification, and secondary authentication. By increasing the effective lifespan of a fingerprint under benign changes (e.g., software updates, configuration shifts), services can maintain continuity of user recognition without over-relying on fragile or user-resettable identifiers like cookies.

However, we acknowledge that improving fingerprint durability also raises privacy concerns. In particular, more stable fingerprints may be exploited to facilitate long-term user tracking without consent, especially in contexts outside security or authentication. In the wrong hands, these techniques could contribute to cross-site tracking, profiling, or surveillance that the user is not aware of.

To mitigate these concerns, we emphasize that this method should only be deployed in trusted, security-sensitive environments where users already expect some form of browser recognition. It should not be used for behavioral advertising or user profiling. Fingerprint histories should be stored securely, with limited retention and scope. Developers and organizations applying this method should ensure transparency and allow users to inspect or reset browser associations when applicable.

Additionally, users who wish to avoid fingerprint-based tracking may opt for browsers that actively interfere with or randomize fingerprint attributes. For example, Brave

TABLE 7. Fingerprint attributes (Part 2).

Attribute	Source	Description	Example
Plugins	JavaScript	Installed browser plugins with MIME info	{ "name": "PDF Viewer", "description": "Portable Document Format", "mimeTypes": ["type": "application/pdf", "suffixes": "pdf" , "type": "text/pdf", "suffixes": "pdf"] }
Languages	JavaScript	Preferred languages list	[en]
Canvas	JavaScript	Encoded image rendered on HTML canvas	"geometry": "..."
Touch support	JavaScript	Touch capability indicators	{"maxTouchPoints": 0, "touchEvent": false, "touchStart": false}
Vendor	JavaScript	Reported browser vendor	"Google Inc."
Vendor flavors	JavaScript	Distinguishes browsers with same engine	chrome
Cookies enabled	JavaScript	Whether cookies are enabled	false
Color gamut	JavaScript	Supported color space	srgb
Inverted colors	JavaScript	Whether display colors are inverted	inverted
Forced colors	JavaScript	Whether a forced color palette is active	false
Monochrome	JavaScript	Whether device uses monochrome display	0
Contrast	JavaScript	Preferred contrast mode	more
Reduced motion	JavaScript	Whether reduced motion is enabled	reduce
HDR	JavaScript	High dynamic range display support	false
Math	JavaScript	Results of floating-point operations	"acos": 1.44..., "acosh": 709...
Video card	JavaScript	GPU vendor and renderer details	"vendor": "Google Inc.", ...
PDFViewerEnabled	JavaScript	Support for in-browser PDF viewing	true

implements fingerprinting defenses by altering or reducing the stability of key attributes [26]. Although we have not empirically tested the effectiveness of these browsers against our algorithm due to their absence in our dataset, their design philosophy aligns with efforts to reduce linkability over time and is therefore expected to reduce ThresholdFP's durability.

We believe that publishing this technique contributes to the broader understanding of browser fingerprinting, both to improve defenses and to inform ethical deployment. Our intention is to support more robust and user-friendly authentication systems, not to enable covert tracking. Continued dialogue among browser developers, privacy advocates, and the research community is essential to ensure fingerprinting technologies are used responsibly.

IX. OPEN SCIENCE & AVAILABILITY

In line with the common practice in the domain, we are unable to share our datasets because of privacy considerations. Additionally, due to contractual obligations, we are unable to share the artifacts.

APPENDIX

Tables 6 and 7 list the fingerprinting attributes used in this study, along with their sources, descriptions, and representative example values.

REFERENCES

[1] U. Iqbal, S. Englehardt, and Z. Shafiq, "Fingerprinting the fingerprinters: Learning to detect browser fingerprinting behaviors," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2021, pp. 1143–1161.

[2] S. Wu, P. Sun, Y. Zhao, and Y. Cao, "Him of many faces: Characterizing billion-scale adversarial and benign browser fingerprints on commercial websites," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2023, pp. 1–16.

[3] H. Jonker, B. Krumnow, and G. Vlot, "Fingerprint surface-based detection of Web bot detectors," in *Proc. 24th Eur. Symp. Res. Comput. Secur. Comput. Secur.* Cham, Switzerland: Springer, Sep. 2019, pp. 586–605.

[4] A. Vastel, W. Rudametkin, R. Rouvoy, and X. Blanc, "FP-crawlers: Studying the resilience of browser fingerprinting to block crawlers," in *Proc. Workshop Meas., Attacks, Defenses Web*, 2020, pp. 1–13.

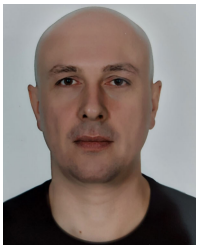
[5] A. Durey, "Leveraging browser fingerprinting to strengthen Web authentication," Ph.D. dissertation, CRIStAL Lab., Inria Centre, Univ. Lille, Lille, France, 2022.

[6] X. Lin, P. Ilia, S. Solanki, and J. Polakis, "Phish in sheep's clothing: Exploring the authentication pitfalls of browser fingerprinting," in *Proc. 31st USENIX Secur. Symp.*, 2022, pp. 1651–1668.

[7] P. Laperdrix, G. Avoine, B. Baudry, and N. Nikiforakis, "Morellian analysis for browsers: Making Web authentication stronger with canvas fingerprinting," in *Proc. Detection Intrusions Malware*, Gothenburg, Sweden, Jun. 2019, pp. 43–66.
[8] A. Vastel, P. Laperdrix, W. Rudametkin, and R. Rouvoy, "FP-STALKER: Tracking browser fingerprint evolutions," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 728–741.
[9] P. Laperdrix, W. Rudametkin, and B. Baudry, "Beauty and the beast: Diverting modern Web browsers to build unique browser fingerprints," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 878–894.
[10] Y. Cao, S. Li, and E. Wijmans, "Cross-browser fingerprinting via OS and hardware level features," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2017, pp. 1–15.
[11] A. Gómez-Boix, P. Laperdrix, and B. Baudry, "Hiding in the crowd: An analysis of the effectiveness of browser fingerprinting at large scale," in *Proc. World Wide Web Conf. World Wide Web*, 2018, pp. 309–318.
[12] P. Eckersley, "How unique is your Web browser?" in *Proc. 10th Int. Symp. Privacy Enhancing Technol.*, Berlin, Germany. Cham, Switzerland: Springer, Jul. 2010, pp. 1–18.
[13] J. R. Mayer, "'Any person. A pamphleteer': Internet anonymity in the age of Web 2.0," Senior thesis, Woodrow Wilson School Public Int. Affairs, Princeton Univ., Princeton, NJ, USA, 2009, vol. 85.
[14] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless monster: Exploring the ecosystem of Web-based device fingerprinting," in *Proc. IEEE Symp. Secur. Privacy*, May 2013, pp. 541–555.
[15] K. Mowery and H. Shacham, "Pixel perfect: Fingerprinting canvas in html5," in *Proc. W2SP*, 2012, pp. 1–12.
[16] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, "The Web never forgets: Persistent tracking mechanisms in the wild," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2014, pp. 674–689.
[17] S. Englehardt and A. Narayanan, "Online tracking: A 1-million-site measurement and analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 1388–1401.
[18] S. Boussaha, L. Hock, M. Bermejo, R. Cuevas Rumin, A. Cuevas Rumin, D. Klein, M. Johns, L. Compagna, D. Antonioli, and T. Barber, "FP-tracer: Fine-grained browser fingerprinting detection via taint-tracking and entropy-based thresholds," *Proc. Privacy Enhancing Technol.*, vol. 2024, no. 3, pp. 540–560, Jul. 2024.
[19] P. Laperdrix, N. Bielova, B. Baudry, and G. Avoine, "Browser fingerprinting: A survey," *ACM Trans. Web (TWEB)*, vol. 14, no. 2, pp. 1–33, 2020.
[20] L. Trampert, D. Weber, L. Gerlach, C. Rossow, and M. Schwarz, "Cascading spy sheets: Exploiting the complexity of modern CSS for email and browser fingerprinting," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*. Reston, VA, USA: Internet Society, 2025, pp. 1–20.
[21] L. Xu, F. Araujo, T. Taylor, J. Jang, and J. Polakis, "Fashion faux pas: Implicit stylistic fingerprints for bypassing browsers' anti-fingerprinting defenses," in *Proc. IEEE Symp. Secur. Privacy (SP)*, Jun. 2023, pp. 987–1004.
[22] P. Laperdrix, O. Starov, Q. Chen, A. Kapravelos, and N. Nikiforakis, "Fingerprinting in style: Detecting browser extensions via injected style sheets," in *Proc. 30th USENIX Secur. Symp.*, 2021, pp. 2507–2524.
[23] N. Takei, T. Saito, K. Takasu, and T. Yamada, "Web browser fingerprinting using only cascading style sheets," in *Proc. 10th Int. Conf. Broadband Wireless Comput., Commun. Appl. (BWCCA)*, Nov. 2015, pp. 57–63.
[24] O. Starov and N. Nikiforakis, "XHOUND: Quantifying the fingerprintability of browser extensions," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 941–956.
[25] K. Solomos, P. Ilia, N. Nikiforakis, and J. Polakis, "Escaping the confines of time: Continuous browser extension fingerprinting through ephemeral modifications," in *Proc. 2022 ACM SIGSAC Conf. Comput. Commun. Secur.*, 2022, pp. 2675–2688.
[26] B. P. Team. (Jun. 2020). *Fingerprint Randomization*. [Online]. Available: https://brave.com/privacy-updates/3-fingerprint-randomization/
[27] S. Li and Y. Cao, "Who touched my browser fingerprint?: A large-scale measurement study and classification of fingerprint dynamics," in *Proc. ACM Internet Meas. Conf.*, Oct. 2020, pp. 370–385.
[28] G. Pugliese, C. Riess, F. Gassmann, and Z. Benenson, "Long-term observation on browser fingerprinting: Users' trackability and perspective," *Proc. Privacy Enhancing Technol.*, vol. 2020, no. 2, pp. 558–577, Apr. 2020.
[29] GitHub. (2024). *Fingerprintjs*. [Online]. Available: https://github.com/fingerprintjs/fingerprintjs
[30] M. D. Network. (2024). *Web Apis*. [Online]. Available: https://developer.mozilla.org/en-U.S./docs/Web/API
[31] J. Wang, W. Zhang, and S. Yuan, "Display advertising with real-time bidding (RTB) and behavioural targeting," *Found. Trends Inf. Retr.*, vol. 11, nos. 4–5, pp. 297–435, 2017.

ELIF ECEM ŞAMLIOĞLU received the B.S. degree in computer science and engineering from Sabancı University, Istanbul, Türkiye, in 2022, where she is currently pursuing the M.S. degree in computer science and engineering.

Her research interests include computer and network security, web privacy, and software engineering.



SINAN EMRE TAŞÇI received the B.S. degree (Hons.) in computer engineering from Marmara University, in 2002, the M.S. degree in electronics engineering and computer science from Sabancı University, in 2006, and the Ph.D. degree in computer engineering from Marmara University, in 2023. Currently, he is the Director of research and development at Fraud.com. His research interests include computer and network security, ad hoc and sensor networks, and game theory with applications to user provided connectivity.

MUHAMMED FATIH GÜLŞEN received the B.Sc. degree in electrical and electronics engineering from Istanbul University. He is currently pursuing the M.Sc. degree in electronics and communication engineering with Yıldız Technical University. He is a Research and Development Machine Learning Engineer at İHS Teknoloji, where he is responsible for end-to-end research and development projects focused on artificial intelligence solutions for the finance sector. His research interests include computer vision, biometrics, identity verification technologies, and federated learning in security applications.

ALBERT LEVI (Senior Member, IEEE) received the Ph.D. degree in computer engineering from Boğaziçi University, Istanbul, Türkiye, in 1999. He is currently a Professor of computer science and engineering with the Faculty of Engineering and Natural Sciences, Sabancı University, Istanbul. Previously, he was a Visiting Faculty Member with the Department of Electrical and Computer Engineering, Oregon State University, and as a Visiting Professor with the Faculty of Computer Science, Dalhousie University. He has authored and co-authored more than 100 papers in refereed journals and conferences. His research interests include computer and network security with emphasis on the IoT, mobile and wireless system security, public key infrastructures (PKI), privacy, and application layer security protocols. He is an Editorial Board Member of IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, *The Computer Journal* (Oxford University Press), and *Computer Networks* (Elsevier). He was a recipient of the 2018 Turkish Informatics Association Prof. Dr. Aydöksal Science Award. He has served on the program committees of various international conferences. He also served as the General and Program Co-Chair for ISCIS 2006, the General Chair for SecureComm 2008, the Technical Program Co-Chair for NTMS 2009, the Publicity Chair for GameSec 2010, the Program Co-Chair for ISCIS 2011, and the Technical Program Area Co-Chair for IEEE CNS 2022.