You Are as You Type: Investigating the Influence of Timestamp Accuracy on the Robustness of Keystroke Biometrics

Florian Dehling*✉, Sebastian Koch*, Luigi Lo Iacono* and Hannes Federrath†
*University of Giessen, Institute of Computer Science, Giessen, Germany
†University of Hamburg, Security in Distributed Systems, Hamburg, Germany
✉Email: florian.dehling@uni-giessen.de

***Abstract*—Keystroke dynamics are behavioral biometric traits that are frequently proposed to be used in novel authentication systems. Keystroke dynamics are based on the analysis of intervals between keystroke events originating from user input and thus directly depend on available timing information. However, modern web browsers limit the accuracy of timestamps to improve security and privacy. This study systematically investigates the impact of limited timestamp accuracies on the performance of keystroke dynamic analysis. By conducting multiple experiments with popular web browsers, we demonstrate that the minor timestamp modifications that mitigate timing side-channel attacks do not interfere with the effectiveness of keystroke dynamics analysis algorithms. Furthermore, they are surprisingly resilient to larger timestamp modifications, which results in a serious threat to users' privacy. This research provides fundamental knowledge enabling researchers, privacy engineers, and browser vendors to study risks in keystroke dynamics-related systems and to develop mitigations against tracking methods that fingerprint users instead of devices or browsers.**

***Index Terms*—keystroke dynamics, timing precision, continuous authentication, user profiling**

I. Introduction

Keystroke Dynamics (KD) represent how individuals use a keyboard to enter text into a digital system. By extracting timing information, such as the duration of a keystroke or the duration between successive keystrokes, behavioral biometric traits are obtained allowing users to be differentiated based on their typing characteristics [1]. KD are mainly used in research into new authentication approaches like continuous authentication [1, 2], with initial applications in real systems [3].

KD-based authentication systems are frequently proposed to secure access to distributed systems, which are usually accessed over the Internet [2, 4]. Thus, web-browser-based implementations are used for both research and practical applications to obtain KD data, which usually includes extracting timing information from JavaScript APIs [5, 6]. Precise timestamps can be misused to perform timing side-channel attacks, in this case especially the microarchitectural attacks Spectre [7] and Meltdown [8], posing severe threats to the security of systems and users. Consequently, as a mitigation, web browser vendors reduced the precision of timing information available via APIs in recent years [9]. Furthermore, timing information available via JavaScript APIs can be used to implement tracking mechanisms related to device fingerprinting [10]. As a consequence, privacy-focused web browsers introduced further strategies to limit the precision of available timestamps to protect users' privacy [11]. However, since KD systems are fundamentally based on the analysis of timing information, knowledge about the relevance of timestamp accuracies for their performance is of utmost importance. This paper is the first to systematically investigate the relationship between the precision of timing information and the effectiveness of KD analysis by conducting a series of experiments, answering the following research questions:

RQ1 How does the security-enhancing browser feature that moderately limits timestamp accuracy to protect against side-channel attacks affect KD analysis algorithm performance?

RQ2 How does the privacy-enhancing browser feature `resistFingerprinting` that substentially limits timestamp accuracy affect KD analysis algorithm performance?

RQ3 Can server-based KD tracking strategies undermine client-based, noise-induced timestamping?

Using two exemplary KD analysis algorithms and associated datasets, we investigate the impact of limited timestamp accuracies induced by security and privacy features of frequently used web browsers. Conducting a series of comparative experiments, we provide the following contributions: (1) We present a comprehensive overview of the accuracy of timestamps available in popular web browsers. (2) We systematically investigate the impact of limiting timestamp precisions on the performance metrics of KD-based authentication and classification systems. (3) We identify serious threats to users' privacy that can emanate from KD data, which may already be extensively collected today.

Summarizing our results, we found that:

- The default strategies for limiting timestamp precisions in web browsers do not have a relevant negative impact on the performance of the analyzed KD algorithms.
- The analyzed KD algorithms can also deal with severe limitations in the precision of timestamps.
- The observed robustness allows potentially sensitive information on typing behavior to be collected without users' knowledge and exploited for privacy-invasive purposes even when protection means are in place.

Preprint, accepted for publication at IEEE TrustCom-2024.

Our work demonstrates that KD can be used in authentication systems even with reduced timestamp accuracy. The investigation of a KD algorithm designed for authentication based on free-text data has shown that a decreasing precision of timestamps has a particular effect on the False Rejection Rate. In contrast, the False Acceptance Rate only increases slowly. This results in the system losing usability rather than security properties, as legitimate users are more likely to be rejected than potential attackers can falsely pass through.

However, the observed robustness has severe implications for user's privacy, as our results also show that it is possible to identify users based on their typing behavior even with low-accuracy data. This reveals a demand for new transparency and intervention measures for users to become aware and enable them to self-protect.

II. Background

In the following we introduce basic foundations for the two core topics of this work: Keystroke Dynamics (KD) and the precision of timing information in web browsers.

A. Keystroke Dynamics

The approach of using typing behavior during users' interaction with a system for authentication purposes has existed since the 1970s [12]. It is based on the observation that the way a keyboard is used is not unique to each individual but differs sufficiently between users within a group to be used as a behavioral biometric trait [1]. Thus, KD are usually not used as a primary authentication factor but as an additional factor for password-based entry-point authentication systems [2] or continuous authentication systems that are deployed in addition to an entry-point authentication mechanism [13].

An individual's typing behavior is characterized by time-based features. Depending on the implementation of a KD analysis algorithm, different time intervals are analyzed, such as intervals between consecutive keystrokes or durations of a key press (from key-down to key-up) relative to a specific key [14]. By extracting these features, models can be created during an enrollment phase representing an individual's typing behavior. These models can be compared against typing samples obtained during the use of an application or logging in to a service to determine if the observed pattern matches the behavior represented in the model.

KD are not limited to a specific type of input device, e.g., a hardware keyboard, but have already been extensively studied for on-screen keyboards of mobile devices' touch screens [15]. Here, KD are often combined with other behavioral biometric data obtained via numerous sensors in smartphones, such as acceleration sensors [16]. However, the performance of pure KD analysis algorithms does not differ between the use of hardware keyboards and on-screen keyboards [17].

Although KD analysis primarily relies on evaluating timing information, the kind of input typed plays an important role as well. Research distinguishes between fixed-text and free-text systems. A fixed-text system is optimized for analyzing constant inputs like passwords or passphrases and generally achieves lower error rates due to consistent keystroke combinations and features [1, 18]. In contrast, a free-text system is not limited by the content of the typed text, dynamically forming extracted features by combinations of key values and timing information based on the pressed keys [19].

The use of KD for authentication systems is still the subject of numerous research projects [3]. Suggested systems can be divided into two use cases: methods analyzing fixed-text inputs to enhance password-based authentication with an additional factor [2, 20] and methods for analyzing free-text data to implement continuous authentication systems that authenticate a user throughout a session [13].

Beyond its use in authentication systems, KD allow an observed typing behavior to be assigned to a user in a classification process [19]. It has been shown that this allows the implementation of an online tracking mechanism to identify users or to derive privacy-sensitive information like, e.g., age and gender [21, 22, 23].

B. Timing Information in Web Browsers

Web browsers offer multiple ways to obtain timing information with varying degrees of accuracy. These include JavaScript APIs like the `Date.now()` function providing milliseconds elapsed since January 1, 1970, UTC [24] and the `performance.now()` function offering a timestamp with a resolution in the microsecond range [25].

In response to timing side-channel attacks that could spy on hardware events or browser behavior solely through a JavaScript program, the precision of the `performance.now()` timer was limited by browser vendors [26]. The discovery of Spectre attacks further limited the precision of timestamps to prevent side-channel attacks executed via JavaScript [7].

Modifications to the `performance.now()` function by different browser manufacturers have continuously changed over the years and particularly vary between browser families [9]. Generally, a two-stage approach is applied to modify timestamps. First, the resolution gets clamped to a target precision. Second, a random jitter of the target accuracy magnitude is applied to prevent obtaining more accurate timing information through interpolation [9].

In most browsers, the precision of timestamps further depends on the origin of a resource. Through site isolation, cross-site resources are processed in a separate rendering process, thus being treated separately from resources originating from the origin website requested by the user [27]. Since cross-site resources potentially pose more risk to users' security and privacy, they are treated in a *non-isolated* mode that offers less precise timing information than resources loaded in *isolated* browsing contexts [25].

Besides the modifications used to prevent timing side-channel attacks, which are enabled by default in all examined web browsers, the Firefox browser offers an additional function to prevent fingerprinting attacks [11]. With device fingerprinting, the analysis of computation times, such as those of the HTML5 Cryptography API [28], can infer the

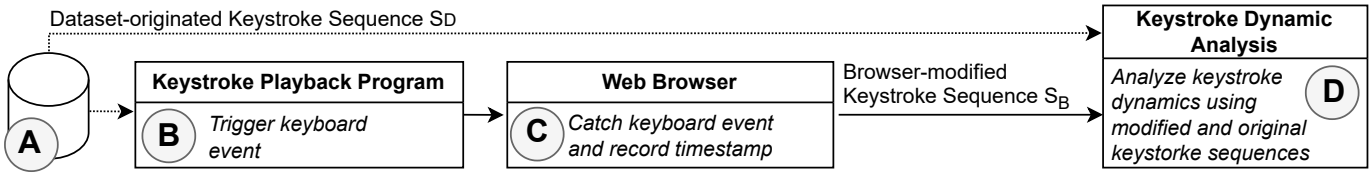


Fig. 1: Schematic structure of the used experimental setup.

used hardware, thus serving as a web tracking tool [10]. The Tor browser, which is based on the Firefox browser, already initially restricts the accuracy of timestamps further to 16.667 ms. For the Firefox browser, this function must be activated via the `privacy.resistFingerprinting` preference, currently without a GUI for end users. Additionally, the target accuracy can be manually selected. Since graphics and video applications in web browsers depend on sufficiently accurate timing information, a default value of 16.667 ms was chosen. This value corresponds to a frame rate of 60 Hz and thus has no impact on displaying media content, while any higher precision limit of general timing information results in drastic limitations on the functionality of web applications that include media content like videos or animations. [29].

In addition to JavaScript functions such as `Date.now()` and `performance.now()`, it is also possible to generate timestamps via indirect methods such as the `SharedArrayBuffer` [30]. However, these are not the subject of the experiments presented here.

III. Experiments Setup

This study investigates the impact of limited timestamp precisions available in web browsers on the performance of KD algorithms for classifying and authenticating users. A comparative analysis is conducted as different web browsers employ different timestamp modification techniques. We decided to examine the popular web browsers of the Chromium browser family (Chrome v114, Chromium v116, Edge v114), Firefox (v114), the Firefox-based Tor Browser (v102), and Safari (v16). Except for Safari (used on MacOS 12.6.4), all browsers were used on Windows 10 (22H2).

Our experimental setup allows to playback and reproduce keystroke sequences from available open datasets and record them within the examined web browsers. The recorded sequences contain the same keystrokes, but the respective timestamps are modified by the used web browser. To assess the effects of these modified timestamps on KD analysis algorithms, the original and the modified sequences are utilized as input data for the KD analysis algorithms, and the resulting error rates are compared. Figure 1 outlines all substeps of our approach described in the following.

Ⓐ We used publicly available KD datasets, including fixed-text and free-text typing samples from multiple users [19, 31]. Each typing sample comprises a series of characters pressed on a hardware keyboard with corresponding timestamps for key-down and key-up events[1].

Ⓑ To simulate user input in real time based on the datasets, we utilized a custom Rust program to trigger keyboard events on the operating system using the Rust library *enigo* [32][2].

Ⓒ For data collection in web browsers, we integrated an HTML input tag on a website. After starting the Rust keystroke simulation, we manually focus on the web page's input field in the browsers and thus use the OS keystroke events to trigger browser keystroke events. We used JavaScript callback functions for key-down and key-up events to retrieve a `DOMHighResTimeStamp` object for each event using the `performance.now()` function. The script for capturing the keystroke events was delivered using a valid HTTPS connection. To measure differences between isolated and non-isolated browsing contexts, the script was loaded as either a same-origin resource (isolated) or from a different domain (non-isolated). In addition to the different web browsers selected for this study, we also repeated the experiments for different browser configurations in the case of Firefox's Resist Fingerprinting function. After capturing a dataset entirely, we exported timestamps along with event types (key-down and key-up) and the corresponding key values.

Ⓓ The browser-modified keystroke sequence S_B served as input for authentication and classification experiments utilizing KD algorithms. To assess the impact of limited timestamp precisions resulting from intercepting keystroke events in web browsers, we examined the performance of the selected algorithms. Therefore, we compared the resulting error rates for wrong classifications or incorrect authentication decisions across different combinations of datasets for the algorithms' input. This includes comparing the algorithms' performance using original (S_D) and modified data (S_B) exclusively, as well as combinations of original and modified data for enrolling and testing the algorithms.

A. Precision of Playback Instrument

We took care to avoid measurement inaccuracies due to the imprecise triggering of keystroke events. We, therefore, compared strategies for delaying program execution and found a busy loop to be the most precise method of triggering events in Rust. This loop queries a high-resolution timestamp from the operating system (using Rust functions from `std::time::Instant`) on each run to check whether the predetermined period has elapsed.

To investigate the accuracy of the employed busy-loop in the Rust-based keystroke playback program, we pro-

[1]The dataset from Gunetti and Picardi [19] contains only key-down events.

[2]The library was customized before use, as additional delays were noticed when running it on MacOS.

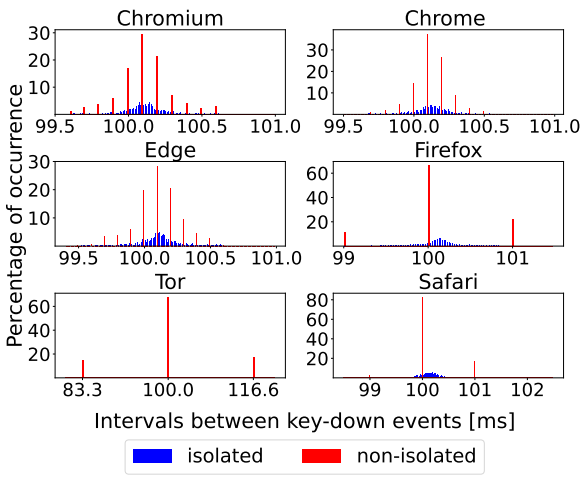


Fig. 2: Histograms for keystrokes recorded in web browsers that were triggered at intervals of 100ms.

grammed the loop to capture a timestamp using the Rust class `std::time::Instant` every 5 ms. In doing so, we collected 10,000 timestamps and calculated the interval of two consecutive records. The resulting average time interval was found to be 5001.6 μs (median 5000.0 μs, standard deviation 99.6 μs), which implies an average error of only $<$2 μs but a comparable high distribution of outliers.

B. Timing Precision in Web Browsers

To evaluate the overall precision of our experiment setup, we used the playback program to trigger 1,000 strokes of a single key in a periodic interval of 100 ms and captured the key-down events in the web browser examined in both isolated and non-isolated mode. As expected, the analysis of the intervals between consecutive key-down events captured in the web browsers shows a dispersion, as can be seen in the histograms (see Figure 2, details in Table I). In addition to the scattering, which can be recognized in both isolated and non-isolated browsing contexts, an offset arises. Except for Safari and the Tor browser, the offset is in the range of [0.10, 0.12] ms, with no systematic difference between isolated and non-isolated browsing modes. For Safari, we measured an average offset of 0.15 ms for isolated and 0.74 ms for non-isolated mode. For the Tor browser, which only supports a non-isolated browsing mode, the offset averaged 0.41 ms. Since KD algorithms are primarily based on analyzing differences in temporal patterns between specific keystroke sequences, a consistent delay applied to every keystroke is not expected to have any effect on the performance.

However, the dispersion of the intervals between two consecutive key-down events indicates additional distortions due to the experimental setup. Obviously, the timestamps recorded in this pre-experiment are already affected by the modifications employed in the web browsers to mitigate timing side-channel attacks (see Sec. II-B). To nevertheless conclude on the precision limitations of our measurement setup, we first used our recordings to confirm the target resolutions R_{target} of the web browsers taken from the browser engine's documentations [33, 34, 35]. We compared the number of distinct interval measurements I per web browser and isolation mode by the number of possible timestamps when considering the target timestamp resolution R_{target} taken from the documentation and the range of divergences $\max(I) - \min(I)$ of the captured intervals I. For all browsers, the observed number of distinct timestamps is smaller or equal to the possible number of timestamps, which allows us to confirm the effects of the mechanisms used in the browser engines (see Table I). Knowing the resolution of timestamps taken from the web browsers further allowed us to interpret the observed standard deviation. Since there are no systematic differences in the standard deviation for conditions with a target timestamp resolution of $\leq$ 0.02 ms (SD [133, 242] μs), we infer that those distortions are rather caused by our experiment setup than by the mechanisms used in the browser engines. For the target accuracy of 1 ms, as found for the Firefox and Safari browsers in non-isolated browsing contexts, the standard deviation rises to 570 μs and 406 μs, respectively.

Based on these observations, we derive that the signal chain in the used experimental setup for this study allows us to conclude on browser-based timestamp manipulations of $\geq$ 1 ms. In fact, considering the distortion of timestamps caused by the used experimental setup does not allow drawing conclusions on timestamp precision limitations $\leq$ 100 μs as applied for Chromium-based browsers or for the Firefox and Safari browsers in isolated browsing contexts. If the experiments with these conditions were to show effects on the performance of the KD analysis algorithms, it would not be possible to differentiate whether these were caused by the procedures used in the browsers to limit the precision of timestamps or whether they were due to the distortions caused by the experimental setup. By conducting comparative experiments with unmodified data for KD analysis algorithms, we can assess the influence of our experimental setup. If no significant effect is observed from the distortions, we conclude that the web browser procedures are also insignificant.

C. Keystroke Dynamics Algorithms

To investigate the impact of browser-induced timestamp smearing on algorithms for KD analysis, we selected exemplary methods for fixed- and free-text, including open datasets from the literature. In addition to sufficiently documented implementations for reproducibility, we gave priority to implementations frequently cited as exemplary candidates in research work rather than focusing on high performance.

1) Fixed-text: For fixed-text analysis, we selected the method proposed by Killourhy and Maxion [31]. The algorithm and the corresponding dataset have been widely used as a benchmark for assessing other KD algorithms [4, 36, 37, 38] and have also been proposed for novel authentication systems [2, 39]. The authors have published an implementation in R and the dataset used for evaluation in their study. The dataset includes typing samples from 51 individuals who each entered

TABLE I: Results for the analysis of the experiment setup precision. Keystrokes are triggered with an interval of 100 ms.

Web Browser	Target accuracy[a] $R_{target}[ms]$	Intervals recorded in the web browsers: Mean[b] $[ms]$	Range[c] $[ms]$	Standard deviation[d] $[ms]$	Number of distinct measurements[e]	Maximum possible number of distinct values[f]
Chromium Isolated	0.005	100.11	1.09	0.174	176	218
Chromium Non-isolated	0.1	100.11	1.30	0.200	14	14
Chrome Isolated	0.005	100.12	0.90	0.155	157	181
Chrome Non-isolated	0.1	100.12	0.90	0.133	10	10
Edge Isolated	0.005	100.11	1.07	0.179	175	214
Edge Non-isolated	0.1	100.10	1.10	0.170	12	12
Firefox Isolated	0.02	100.11	1.52	0.242	75	77
Firefox Non-isolate	1.0	100.11	2.00	0.570	3	3
Safari Isolated	0.02	100.15	1.02	0.172	50	52
Safari Non-isolated	1.0	100.74	2.00	0.406	3	3
Tor	16.667	100.41	33.33	9.468	3	3

[a] According to source codes form Chromium [33], Safari [34], and Firefox [35].
[b] Average of intervals recorded in the web browsers. [c] Range of intervals I recorded in the web browsers: $max(I) - min(I)$.
[d] Standard deviation of intervals I recorded in the web browsers. [e] Number of distinct measurements of intervals I recorded in the web browsers.
[f] Maximum number of distinct values considering the range of intervals I recorded and the target accuracy R_{target}: $((max(I) - min(I)/R_{target}) + 1$

the password `.tie5Roanl` 400 times. The accuracy of the study environment for collecting the timestamps of keyboard events is reported as ±200 μs.

We translated the R script provided by the researchers into Python, including anomaly detection algorithms based on the Euclidean [40], Manhattan [41], and Mahalanobis distances [41]. We furthermore implemented the Scaled Manhattan distance [42] following the description in the publication, as it proved to perform best [31].

Our experiment for testing the algorithm's performance was carried out as follows. For enrollment, each individual's first 200 password entries are used as a profile representing their normal behavior. The remaining 200 entries of the password are then used to test the system, representing the keystrokes of a legitimate user. Furthermore, the first 50 entries of each remaining subject are used to attack a profile. Since the authors provided the same dataset as used in their study, we were able to validate our implementations. Reproducing the experiments yielded the same results.

2) Free-text: For free-text keystroke analysis, we selected the algorithm presented by Gunetti and Picardi [19]. The decisive factors were the detailed documentation of the applied algorithms and the conducted experiments, the differentiation between classification and authentication tasks, and the utilization of the available dataset and the proposed algorithms in further studies [4, 43, 44, 45].

In the algorithms for classification and authentication, two main distance measures are employed: (1) R-distance measures the rank ordering of time intervals between letter sequences to characterize a user's typing behavior. It compares typing samples by counting discrepancies in these ranked intervals. (2) A-distance assesses the absolute times between keystrokes of letter sequences and compares them across samples. A ratio is calculated and compared against a threshold to determine similarity.

The publication's original dataset was unavailable due to changes in data protection law. Nevertheless, the authors could provide us with data from 31 users, each providing 15 Italian text samples. The data was collected through a web application. The subjects providing the text samples were not given any specific instructions regarding the text's content and used their own devices for entering text. The web browsers used to gather the samples are not known. The resolution of the collected timestamps is 10 ms [19]. The dataset was published in 2005, and web browsers introduced timestamp precision limitations in 2015 [9]. Therefore, we assume that no additional modifications of timing information comparable to those included in today's web browsers happened at the time of collecting the data. The algorithm allows the combination of A- and R-distances for multiple lengths of keystroke sequences (n-graphs). We limit our evaluations to the combination R_{234}+A_{23} since it performed best with the unmodified dataset. Special to the selected algorithm is that it allows for both classification—identifying the closest user profile—and authentication—confirming that a sample matches the classified profile close enough, considering that it could originate from an unknown user. Since the classification approach allows a typing sample to be assigned to an individual from a set of users, which corresponds to the use case of tracking, we use this algorithm to assess users' privacy risks.

We replicated the experiments from Gunetti and Picardi [19] with small modifications. In addition to the smaller dataset, we also used only samples of legitimate users as attackers on other profiles to test the authentication algorithm. This leads to fewer simulated attacks than in the original publication, which could result in better performance results. Our first run revealed unusually poor results for text samples from three out of the 31 participants. An examination of the respective data revealed an inconsistent use of lower- and uppercase letters and an extensive use of special characters. Excluding those leads to performance results comparable to those in the original publication [19]. We have, therefore, refrained from further optimizing the data's pre-processing and have excluded the three datasets in question, resulting in a total size of 28 users.

IV. Experiments

Below, we present three experiments carried out to investigate the impact of limited timestamp precisions on the performance of KD analysis algorithms. First, we analyze the influence of default timestamp precision used to mitigate timing side-channel attacks to answer RQ1. Second, referring to RQ2, we investigate the effect of the privacy-preserving Firefox function Resist Fingerprinting (see Sec. IV-B). The third experiment examines how a server-side collection of timestamps impacts the performance of a KD classification algorithm to study privacy implications under RQ3 (see Sec. IV-C).

A. RQ1: Default Browser Settings

1) Experiment Design: We first investigated how web browsers' timestamp precision limitations, commonly employed to mitigate timing side-channel attacks, affect the performance of the selected algorithms for fixed-text and free-text analysis. The approach was the same for both algorithms. The keystroke playback program triggered keystrokes according to the respective datasets. Key-down and key-up events were intercepted in the web browsers, and the timestamps were recorded using the `performance.now()` JavaScript function. For all web browsers (except for the Tor browser), the script for recording keyboard events was loaded as same origin for the isolated mode and as a resource from a different domain for the non-isolated mode. We configured all web browsers with default settings. A warm-up phase consisting of eight keystrokes was executed before each input (i.e., before each passphrase from the fixed-text dataset or before each of the text samples of a user from the free-text dataset) to avoid potential impacts from memory allocation or similar. The simulation execution time was 23 hours for the fixed-text dataset, consisting of 400 password entries each from 51 users, and 40 hours for the free-text dataset, comprising 15 typing samples each from 28 users.

2) Results: We used data collected in the experiments as input for the KD algorithms presented in Sec. III-C. We utilized different combinations of original and modified datasets to enroll keystroke profiles and test the authentication and classification performance. These include using original data for enrollment and modified data for testing (Orig-Mod), modified data for enrollment and original data for testing (Mod-Orig), and modified data for enrollment and testing (Mod-Mod).

a) Fixed-text: Following the work of Killourhy and Maxion [31], we report the average Equal Error Rates (EER), which is the operation point of the algorithm where the error rates for false positive and false negative decisions are equal (see Table II). Overall, no systematic effect of the limited timestamp precision from the web browsers can be identified for any of the analyzed combinations of enrollment and test data. Furthermore, comparing the EER achieved with the original data allows us to conclude that the distortion of the timestamps caused by the experimental setup used has no influence on the fixed-text experiments.

Differentiating between the various distances used in the algorithm indicates that the Mahalanobis distance is more sensitive to timestamp inaccuracies than the others. Although the results for the Tor browser show a slight increase in EER for the Manhattan Scaled distance for the combinations Mod-Mod and Orig-Mod, we cannot derive a systematic effect from these individual observations due to the minor deviations.

b) Free-text: Similar to the fixed-text experiment, we differentiated between original and modified data combinations for enrollment and testing. Furthermore, we divided the evaluation into results for authentication and classification based on text samples. Following the reporting of results by Gunetti and Picardi [19], we report the False Acceptance Rate (FAR) and the False Rejection Rate (FRR) for authentication, and the Misclassification Rate (McR) for the classification experiment.

Except for timestamps recorded with the Tor browser and used for both enrollment and testing, we observed no increase of error rates (see Table III). Instead, the distortion of timestamps by the experimental environment apparently led to a slight improvement in error rates. However, due to the very small deviations, we do not consider this observation to be relevant in practice.

Answering RQ1, we found that default timestamp modifications for timing side-channel attack mitigation have no significant impact on the performance of the investigated fixed-text and free-text KD analysis algorithms.

B. RQ2: Fingerprinting Mitigation

In a second experiment, we investigated the effects of the Firefox browser's privacy-enhancing anti-tracking function Resist Fingerprinting (RFP). Unlike the measures against timing side-channel attacks previously examined, at the time of conducting the experiments, the RFP function needs to be explicitly enabled using the setting `privacy.resistFingerprinting` in the browser's *about:config* preferences. Initially, enabling it limits the precision of timestamps to 16.667 ms, corresponding to the default settings of the Tor browser. However, this value can be adjusted using a preference attribute[3], which was utilized for the experiment described below.

1) Experiment Design: To examine the impact of different configurations of the RFP function, we largely replicated the protocol described in Sec. IV-A. Differences arose due to the limitation of the investigation of the Firefox browser only and the variability of the target precision of timestamps. In addition to the initial setting of 16.667 ms, configurations with intervals of 20 ms ranging from 20 ms to 200 ms were explored. The experiments were conducted entirely for each of the 11 settings, resulting in 22 simulation runs (i.e., 11 runs for the free-text dataset and 11 runs for the fixed-text dataset).

2) Results: Similar to the preceding experiment, we utilized different combinations of modified data and original data for enrollment and testing across various RFP settings.

[3] privacy.resistFingerprinting.reduceTimerPrecision.microseconds

TABLE II: Average Equal Error Rates (%) of fixed-text keystroke dynamics analysis with timestamp precisions limited by web browsers with default settings. Performance with original data only: Euc: 17%, Man: 15%, ManS: 10%, Mah: 11%.

		Mod - Mod				**Mod - Orig**				**Orig - Mod**			
		Euc	Man	ManS	Mah	Euc	Man	ManS	Mah	Euc	Man	ManS	Mah
Chromium	I	17	15	10	12	17	15	10	09	17	15	10	11
	NI	17	15	10	11	17	15	10	11	17	15	10	11
Chrome	I	17	15	10	11	17	15	10	09	17	15	10	11
	NI	17	15	10	11	17	15	10	11	17	15	10	11
Edge	I	17	15	10	12	17	15	10	09	17	15	10	11
	NI	17	15	10	12	17	15	10	09	17	15	10	11
Firefox	I	17	15	10	12	17	15	10	09	17	15	10	11
	NI	17	15	10	11	17	15	10	11	17	15	10	11
Safari	I	17	15	10	12	17	15	10	09	17	15	10	11
	NI	17	15	10	11	17	15	10	11	17	15	10	11
Tor	NI	17	15	11	13	17	15	10	08	17	16	11	14

Mod - Mod: Modified data for enrollment and testing. **Mod-Orig**: Modified data for enrollment, original data for testing.
Orig-Mod: Original data for enrollment, modified data for testing. **Euc**: Euclidean, **Man**: Manhattan, **ManS**: Manhattan Scaled, **Mah**: Mahalanobis
I: Isolated browsing mode, **NI**: Non-isolated brosing mode

TABLE III: Error rates of free-text keystroke dynamics analysis with timestamps precisions limited by web browsers with default settings. Results with original data only: Misclassification Rate 2.96%, False Rejection Rate: 7.14%.

		Misclassification Rate [%]			False Rejection Rate [%]			False Acceptance Rate [%]		
		Mod-Mod	**Mod-Orig**	**Orig-Mod**	**Mod-Mod**	**Mod-Orig**	**Orig-Mod**	**Mod-Mod**	**Mod-Orig**	**Orig-Mod**
Chromium	I	2.22	2.22	2.47	6.19	5.95	5.24	0.00	0.00	0.00
	NI	2.47	2.47	2.47	6.43	5.71	5.24	0.00	0.00	0.00
Chrome	I	2.22	2.22	2.47	5.71	5.71	5.24	0.00	0.00	0.00
	NI	2.22	2.22	2.47	6.43	5.71	5.24	0.00	0.00	0.00
Edge	I	2.22	2.22	2.47	5.95	5.71	5.00	0.00	0.00	0.00
	NI	2.22	2.22	2.47	6.43	5.71	5.48	0.00	0.00	0.00
Firefox	I	2.47	2.47	2.47	6.19	5.71	5.00	0.00	0.00	0.00
	NI	2.22	2.22	2.72	6.43	6.67	6.20	0.00	0.00	0.01
Safari	I	2.22	2.22	2.47	6.19	5.71	5.00	0.00	0.00	0.00
	NI	2.22	2.22	2.47	6.43	6.67	5.95	0.00	0.00	0.00
Tor	NI	3.21	2.47	2.22	7.38	4.52	5.48	0.00	0.00	0.00

Mod - Mod: Modified data for enrollment and testing. **Mod-Orig**: modified data for enrollment, original data for testing.
Orig-Mod: original data for enrollment, modified data for testing. **I**: Isolated browsing mode, **NI**: Non-isolated browsing mode

a) Fixed-text: We found that for fixed-text analysis, the combination of original data for enrollment and RFP-modified data for testing results in a drastic increase of average EER for the Mahalanobis and the Manhattan Scaled distance with EER of 91% and 77% respectively for RFP at 200 ms (see Figure 3, details in Appendix Table V). Using the same combination of datasets, the Euclidean and Manhattan distances were shown to be more robust, resulting in a maximum EER of 33% and 46%. If the RFP-modified data is also used for enrollment, the EER of different distances behaves more similarly. Enrolling the keystroke profiles with RFP-manipulated data and testing the algorithm with unmodified original data results in a slight performance increase, even for RFP at 200 ms. We deduce from this that the variances of the timestamps resulting from the manipulation experiments are still smaller than the variance required to differentiate individuals based on typing behavior.

b) Free-text: Similar to the fixed-text experiment, the performance of the free-text-based authentication algorithm for higher RFP settings depends on the combination of enrollment and testing data and shows the most negative impact when enrolling the profiles with original data (see Figure 4a). If the enrollment is done with RFP-modified data, the FRR rises to 32% if tested with RFP-modified data and 17% when tested with unmodified data. Notably, while the FRR rises, the FAR, which is the rate at which the system mistakenly classifies an attacker as a legitimate user, remains below 0.3% in all cases (see Appendix, Table VI). This is due to the algorithm's additional verification step during the authentication process, checking whether the resulting distance measures fall within the magnitude of distance measures stored in the profile.

Using only RFP-manipulated data, the McR for the free-text classification algorithm rises up to 24% (see Figure 4b). Enrolling with original data does not significantly reduce performance. Error rates for both authentication and classification typically increase with RFP settings above 20 ms.

For RQ2, we discovered that reducing timestamp precision via Firefox's RFP function significantly hinders authentication algorithm performance when manipulated timestamps are used for testing but not during keystroke profile enrollment. Using RFP-modified data only, the EER for the fixed-text algorithm only decreases by up to 10% while the FRR of the free-text algorithm increases by 25%, both if timestamp precision is limited to 200 ms. Since the impact on the FAR for free-text experiments is minimal, the security of an authentication system would not be significantly affected.

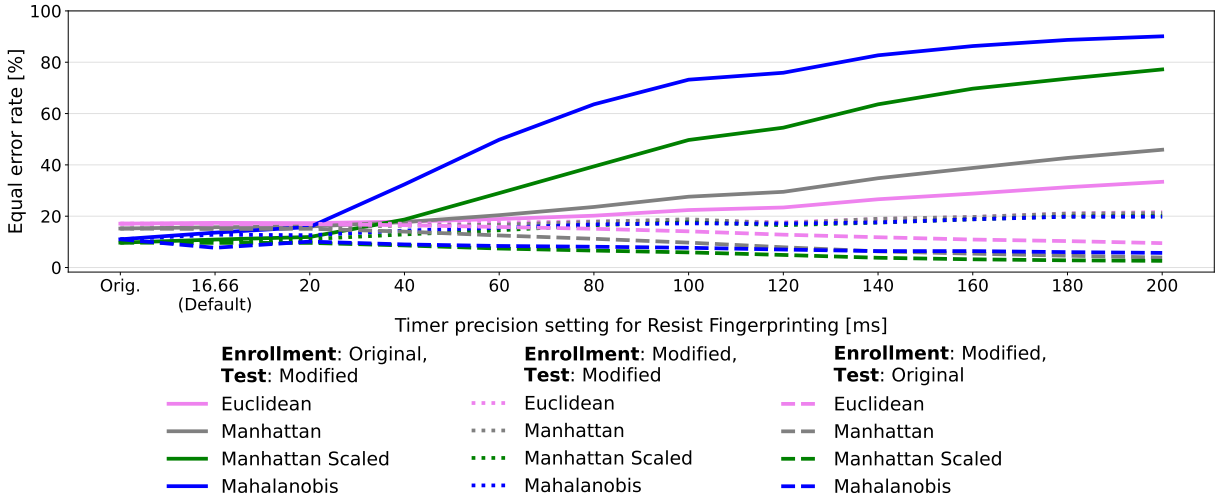
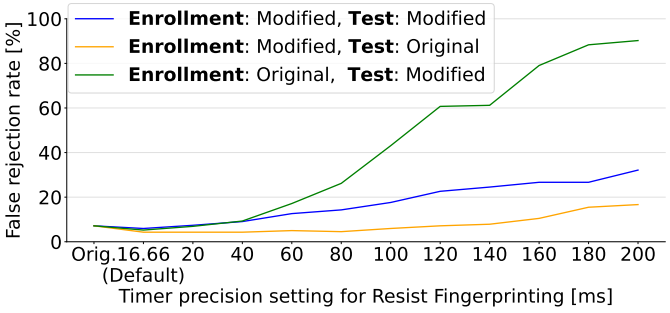
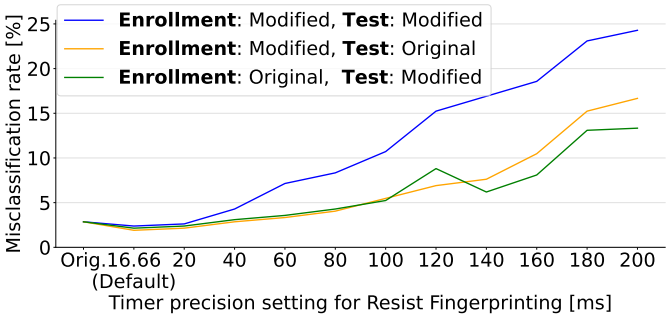


Fig. 3: Average Equal Error Rates for the fixed-text experiment with activated Resist Fingerprinting function in Firefox.



(a) False Rejection Rates



(b) Misclassification Rates

Fig. 4: Error rates for free-text experiments with timestamps modified by the Firefox Resist Fingerprinting function.

C. RQ3: Remote Capturing of Keystroke Events

The moderate increase in misclassification rates for timestamps modified by the Resist Fingerprinting feature for the classification task raises questions on protecting user's privacy considering the use of KD for user tracking [23]. A possible countermeasure could be extended functions of web browsers that either search for suspicious tracking scripts using static analyses or allow the use of precise timestamps to be controlled via user approval [9]. However, client-based solutions would only make sense if tracking entities would have no other way to gain sufficiently precise KD data. We, therefore, performed a follow-up experiment to investigate the feasibility of obtaining timestamps for KD remotely on a server.

1) Experiment Design: Contrary to the previous experiments, instead of using the key-down JavaScript callback to capture a timestamp, we send the value of the pressed key to a server using either WebSocket messages or HTTP requests. Once the value arrives on the server, a timestamp is obtained and stored. The approach to play back the keystrokes is the same as in the previous experiments. Given our primary focus on privacy implications, we have narrowed the scope of this experiment to the classification algorithm and the free-text dataset from Gunetti and Picardi [19]. The classification approach is of significant interest as it could be used to implement a user-tracking mechanism based on KD, thereby highlighting the crucial privacy implications.

Our experiment involved two settings: data transmission via WebSocket messages and HTTP. Each setting was further diversified by using two server locations. The client computer, situated in Germany for both conditions, interacted with servers located in Germany and the US (AWS region North America). To ensure consistency, we used a Chromium web browser with default configuration, secured all transmissions with TLS, and loaded the JavaScript to catch the key-down events as an isolated resource. We deployed a Rust program on the server to receive WebSocket and HTTP messages and to obtain a timestamp using the `std::time::instant` class. For each of the four transmission protocol and server location combinations, we replayed the complete free-text data set consisting of 15 text samples from 28 users.

2) Results: We measured round trip times (RTT) to infer the connection quality in parallel with replaying and transmitting the keystrokes. For experiments involving WebSocket connections, ping messages were sent every 4 seconds. In experiments with HTTP transmission, the RTT was documented

TABLE IV: Results of the classification experiment with free-text data and timestamps recorded on a web server after character-wise transmission of key-down events. The client is located in Germany. The misclassification rate with unmodified timestamps is 2.86 %.

Server location	Transport Protocol	RTT $[ms]$ Mean	RTT $[ms]$ SD	Misclassification Rate [%]
Germany	HTTPS	23	6	2.38
Germany	WSS	19	5	2.14
USA	HTTPS	168	58	2.14
USA	WSS	158	16	2.14

using corresponding HTTP requests every 2 seconds.

Following our previous results, the standard deviation of RTTs $\leq$ 58 ms already indicates that even a transmission of single characters using HTTP messages to a server located on a different continent does not affect the precision of a server-sided gained timestamp used for classifying users based on their KD (see Table IV). Instead, a slight improvement from 2.86% to as low as 2.14% was observed. This is consistent with previous findings, wherein minor modifications led to better performance metrics. Additionally, it is important to note that in this experiment, the distortions of original timestamps do not occur systematically but depend on the quality of the connection at the moment of data transmission.

Answering RQ3, our investigation revealed that timestamps acquired on a server following the character-wise transmission of user keystrokes over the Internet still enable the classification of users based on their KD without adversely affecting the algorithm's performance. Consequently, we assert that constraining timestamp precision in web browsers is inadequate for safeguarding users' privacy with regard to online tracking predicated on KD.

V. Discussion

Conducting multiple experiments, we investigated how limited timestamp precisions caused by timing side-channel attack prevention and device fingerprinting mitigation strategies of web browsers affect the performance of KD analysis algorithms. We demonstrate that modifications of timestamps in the range of <1 ms, as used for timing side-channel attack mitigation, do not negatively impact the error rates of the algorithms studied. With more significant modifications, i.e., limiting the precision of timestamps up to 200 ms, the investigated algorithms show surprisingly robust responses under certain conditions. We found that for authentication purposes, error rates only rise drastically when a KD profile is enrolled with more precise data than is used later for authentication. Practically, this implies that limitations in the functionality of KD systems are particularly to be expected when the resolution of available timestamps decreases while existing reference profiles are still created with higher resolution timing information. This is only to be expected in special cases, such as when a web browser update introduces a significant limitation of timestamp precision or a user switches between web browsers with very different timestamp accuracies.

Our experiments further reveal that when data with limited precision is already used for profile enrollment, the algorithms still perform adequately with lower-detailed temporal information. Impacts on performance metrics were observed starting with timestamp precisions between 20 ms and 40 ms. We conclude that the temporal variance that characterizes users' typing behavior is >20 ms and, in parts, even >200 ms, which is why manipulating the time information in this area has only a limited influence on the performance of the tested KD algorithms. Since this is solely due to the nature of human typing behavior, we suspect the robustness to be evident in other KD algorithms not tested in this work, too.

Our results indicate that KD-based authentication systems are robust against limited timestamp precisions, which implies serious threats to users' privacy. Our experiments with the free-text-based KD classification algorithm reveal that keystroke samples can be assigned to individuals even with low precise timestamps. We further demonstrate that timestamps obtained on a server after character-wise transmission of key-down events—even by using a single HTTP request per keystroke and long transmission routes—do not decrease the performance of the classification algorithm examined at all.

Considering that it has already been demonstrated that personal data such as age or gender can be derived from KD [21], as well as its suitability as a feature for online tracking [23], this finding emphasizes the serious implications for the privacy of users. Since such a tracking system would not rely on executing code in users' web browsers to obtain precise timestamps, such an approach could virtually not be prevented nor detected. Furthermore, the character-wise transmission of keystrokes is not uncommon, particularly in search fields offering suggestions through a search-as-you-type feature, where user inputs are immediately sent character by character to a dedicated service [46]. Thus, typing profile data for enrollment could already be gathered without user awareness, whether intentionally or unintentionally, such as through log files recording search function requests.

This easy accessibility of KD data thus already poses a privacy risk for users. Furthermore, utilizing KD in authentication systems poses security risks, as concealed collected KD data can compromise KD-based authentication systems through impersonation attacks [47].

Our study suggests that existing strategies for mitigating device fingerprinting attacks are ineffective in preventing *user fingerprinting* based on KD. Reducing the precision of timestamps obtained via web browser APIs does not suffice, as our findings demonstrate that server-side capture of KD, without reliance on client-side timestamps, is feasible. To counter such remote KD capture, interventions should involve direct modification of keystroke event handling, as opposed to reliance on timestamp retrieval APIs. However, such implementation can be problematic, as manipulating events could impact the user interface, negatively affecting usability. Subsequent research could explore more active keystroke obfuscating strategies, in

which web browsers simulate synthetic typing behavior and thus limit the number of individual strokes.

Despite robustness against timestamp modifications, our results indicate that using KD-based authentication systems is not advisable due to the impossibility of maintaining typing profiles confidentiality with today's web browsers.

VI. Related Work

The following summarises work closely related to our experiments. Foundations about the application areas of KD and web browser timers are located in Sec. II.

To our knowledge, the impact of browser-induced timestamp modifications on KD analysis algorithms for both authentication and classification purposes has not previously been studied. However, previous work has analyzed isolated subareas of this complex. Rokicki et al. [9] surveyed the evolution of timer precisions in JavaScript in the context of microarchitectural side-channel attacks. They investigated the mechanisms applied to reduce timer precision in Firefox and Chrome browsers, their evolution over time, and their implications for timing side-channel attacks. Our work updates the information on the accuracy of the available timing information and extends it by analyzing additional web browsers.

The impact of manipulated timing information on KD has been studied in the context of anonymization approaches to prevent the identification of subjects in open KD datasets and impersonation attacks. Leinonen et al. [48] proposed rounding and bucketing timestamps for KD from an online programming class. They conclude that rounding to the nearest 500 ms or bucketing timestamps to even-sized 300 ms buckets results in the best anonymization when using a KD analysis algorithm that is based on digraphs only. Monaco and Tappert [49] investigated obfuscation strategies based on delaying keystroke events and tested their mechanisms with different fixed- and free-text datasets. They conclude that a 25 ms delay reduces the identification accuracy by 20%, while 500 ms is required to half the accuracy.

In contrast to previous studies on obfuscating KD, our work considers the differentiation of authentication and classification tasks. Furthermore, the KD algorithms selected for this work generally perform better and seem more robust than the classifiers used in related work. Facing the potential privacy risks arising from the misuse of KD, further enhancing obfuscation strategies is required.

VII. Limitations

This study faces several limitations related to its experimental setup, as detailed in Sec. III. The primary issue is the limited precision due to additional distortions through event handling between the keystroke playback program and the web browsers. In fact, the setup used does not allow us to conclude about the limitations of timestamp precisions as used in web browsers offering an isolated browsing context. However, since we did not find an impact on the performance of the examined KD analysis algorithms, even with bigger timestamp manipulations as caused by our experiment setup (see Sec. IV-A), we conclude that the precision of the environment used is sufficient for the experiments conducted in this study.

Our experiments are limited to two KD algorithms and their datasets. Therefore, the absolute measurement results can not be generalized. However, given that our results show that the default manipulations in web browsers result in less temporal variance than needed to attribute typing behavior to a user, we do not expect the basic performance of other algorithms to provide further insights into the influence of modified timestamps. Since the selected free-text algorithm did not show any performance loss in the experiment with server-side generated typing profiles, our answer to research question RQ3 would remain the same even with more or less robust additional KD algorithms. The limited number of subjects included in the datasets prevents broader generalizations to larger systems. However, scalability is a fundamental issue of KD algorithms, which we do not consider in this paper because performance changes could then no longer be attributed solely to the accuracy of the timing information used.

Finally, the target precisions for timestamps used in web browsers have changed in the last years [9]. Our observations can only be related to the browser versions used, as it is to be expected that other precision limits will be used in response to new security and privacy vulnerabilities.

VIII. Conclusion and Outlook

In this study, we conducted three experiments to explore the impact of limited timestamp precisions on the performance of KD analysis algorithms utilized for user authentication and classification. Our findings indicate that the current security and privacy measures implemented in web browsers, which restrict the precision of timestamps available through JavaScript APIs, do not adversely affect the performance of the KD analysis algorithms under investigation. Furthermore, we observed that more stringent limitations on timestamps within web browsers are ineffective in safeguarding users' privacy from potential invasions by KD-based tracking systems, as server-side retrieval of sufficiently precise KD data can bypass the timestamp modifications implemented in current web browsers. Our results emphasize the need for future research to concentrate on developing more robust strategies for obscuring KD in web browsers and to evaluate the current extent of KD data collection, which may already facilitate the creation of typing profiles during online service usage.

Acknowledgments

We want to thank Claudia Picardi and Daniele Gunetti for providing the data set and supporting the implementation of the algorithms. Special thanks go to Martha Lacek for her essential support in the early stages of this work.

References

[1] H. Crawford, "Keystroke dynamics: Characteristics and opportunities," in *2010 Eighth International Conference on Privacy, Security and Trust*. Ottawa, Canada: IEEE, Aug. 2010, pp. 205–212.

[2] A. A. Wahab, D. Hou, and S. Schuckers, "A User Study of Keystroke Dynamics as Second Factor in Web MFA," in *Proceedings of the Thirteenth ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '23. New York, NY, USA: Association for Computing Machinery, Apr. 2023, pp. 61–72.

[3] S. Roy, J. Pradhan, A. Kumar, D. R. D. Adhikary, U. Roy, D. Sinha, and R. K. Pal, "A Systematic Literature Review on Latest Keystroke Dynamics Based Models," *IEEE Access*, vol. 10, pp. 92 192–92 236, 2022.

[4] A. A. Wahab, D. Hou, S. Schuckers, and A. Barbir, "Securing Account Recovery Mechanism on Desktop Computers and Mobile Phones with Keystroke Dynamics," *SN Computer Science*, vol. 3, no. 5, p. 360, Jul. 2022.

[5] R. Giot, M. El-Abed, and C. Rosenberger, "Web-Based Benchmark for Keystroke Dynamics Biometric Systems: A Statistical Analysis," in *2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, Jul. 2012, pp. 11–15.

[6] I. Traore, I. Woungang, M. S. Obaidat, Y. Nakkabi, and I. Lai, "Combining Mouse and Keystroke Dynamics Biometrics for Risk-Based Authentication in Web Environments," in *2012 Fourth International Conference on Digital Home*, Nov. 2012, pp. 138–145.

[7] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre Attacks: Exploiting Speculative Execution," in *2019 IEEE Symposium on Security and Privacy (SP)*. San Francisco, CA, USA: IEEE, May 2019, pp. 1–19.

[8] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, M. Hamburg, and R. Strackx, "Meltdown: reading kernel memory from user space," *Commun. ACM*, vol. 63, no. 6, p. 46–56, May 2020.

[9] T. Rokicki, C. Maurice, and P. Laperdrix, "SoK: In Search of Lost Time: A Review of JavaScript Timers in Browsers," in *2021 IEEE European Symposium on Security and Privacy*, Sep. 2021, pp. 472–486.

[10] I. Sanchez-Rola, I. Santos, and D. Balzarotti, "Clock Around the Clock: Time-Based Device Fingerprinting," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. Toronto Canada: ACM, Oct. 2018, pp. 1502–1514.

[11] "Mozilla wiki - Privacy Task Force/firefox about config privacy tweeks," 2018, accessed: 2024-04-16. [Online]. Available: https://wiki.mozilla.org/Privacy/Privacy_Task_Force/firefox_about_config_privacy_tweeks

[12] R. Spillane, "Keyboard apparatus for personal identification," *IBM Technical Disclosure Bulletin*, vol. 17, p. 3346, 1975.

[13] I. Stylios, A. Skalkos, S. Kokolakis, and M. Karyda, "BioPrivacy: Development of a Keystroke Dynamics Continuous Authentication System," in *Computer Security. ESORICS 2021 International Workshops*. Cham: Springer International Publishing, 2022, pp. 158–170.

[14] P. H. Pisani and A. C. Lorena, "A systematic review on keystroke dynamics," *Journal of the Brazilian Computer Society*, vol. 19, no. 4, pp. 573–587, Nov. 2013.

[15] E. Maiorana, H. Kalita, and P. Campisi, "Mobile keystroke dynamics for biometric recognition: An overview," *IET Biometrics*, vol. 10, no. 1, pp. 1–23, 2021.

[16] C. Giuffrida, K. Majdanik, M. Conti, and H. Bos, "I Sensed It Was You: Authenticating Mobile Users with Sensor-Enhanced Keystroke Dynamics," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, S. Dietrich, Ed. Cham: Springer International Publishing, 2014, pp. 92–111.

[17] A. K. Belman and V. V. Phoha, "DoubleType: Authentication Using Relationship Between Typing Behavior on Multiple Devices," in *2020 International Conference on Artificial Intelligence and Signal Processing (AISP)*. Amaravati, India: IEEE, Jan. 2020, pp. 1–6.

[18] S. S. Zeid, R. A. ElKamar, and S. I. Hassan, "Fixed-Text vs. Free-Text Keystroke Dynamics for User Authentication," *Engineering Research Journal - Faculty of Engineering*, vol. 51, no. 1, pp. 95–104, Jan. 2022.

[19] D. Gunetti and C. Picardi, "Keystroke analysis of free text," *ACM Transactions on Information and System Security*, vol. 8, no. 3, pp. 312–347, Aug. 2005.

[20] Y. Patel, K. Ouazzane, V. T. Vassilev, I. Faruqi, and G. L. Walker, "Keystroke Dynamics using Auto Encoders," in *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, Jun. 2019, pp. 1–8.

[21] I. Tsimperidis and A. Arampatzis, "The Keyboard Knows About You: Revealing User Characteristics via Keystroke Dynamics," *International Journal of Technoethics (IJT)*, vol. 11, no. 2, pp. 34–51, Jul. 2020.

[22] A. Pentel, "Predicting User Age by Keystroke Dynamics," in *Artificial Intelligence and Algorithms in Intelligent Systems*, R. Silhavy, Ed. Cham: Springer International Publishing, 2019, pp. 336–343.

[23] P. Chairunnanda, N. Pham, and U. Hengartner, "Privacy: Gone with the Typing! Identifying Web Users by Their Typing Patterns," in *2011 IEEE Third Int'l Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third Int'l Conference on Social Computing*. Boston, MA, USA: IEEE, Oct. 2011, pp. 974–980.

[24] "Date.now() - JavaScript | MDN," Nov. 2023, accessed: 2024-04-16. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/now

[25] "Performance: now() method - Web APIs | MDN," Apr. 2023, accessed: 2024-04-16. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Performance/now

[26] Y. Oren, V. P. Kemerlis, S. Sethumadhavan, and A. D. Keromytis, "The Spy in the Sandbox: Practical Cache Attacks in JavaScript and their Implications," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer*

and Communications Security. Denver Colorado USA: ACM, Oct. 2015, pp. 1406–1418.

[27] C. Reis, A. Moshchuk, and N. Oskov, "Site isolation: Process separation for web sites within the browser," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 1661–1678.

[28] "Web Crypto API - Web APIs | MDN," May 2024, accessed: 2024-05-22. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Web_Crypto_API

[29] "Bugzilla - resist-fingerprinting makes it hard to have smooth (60hz) animations," 2021, accessed: 2024-04-16. [Online]. Available: https://bugzilla.mozilla.org/show_bug.cgi?id=1692609

[30] M. Schwarz, C. Maurice, D. Gruss, and S. Mangard, "Fantastic Timers and Where to Find Them: High-Resolution Microarchitectural Attacks in JavaScript," in *Financial Cryptography and Data Security*, A. Kiayias, Ed. Cham: Springer International Publishing, 2017, vol. 10322, pp. 247–267.

[31] K. S. Killourhy and R. A. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics," in *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*. Lisbon, Portugal: IEEE, Jun. 2009, pp. 125–134.

[32] "enigo - crates.io: Rust Package Registry," Apr. 2023, accessed: 2024-04-19. [Online]. Available: https://crates.io/crates/enigo/0.1.2

[33] "time_clamper.h in chromium sourcecode," 2022, accessed: 2024-04-19. [Online]. Available: https://source.chromium.org/chromium/chromium/src/+/main:third_party/blink/renderer/core/timing/time_clamper.h;drc=9e110c64fee9c391021dc81944912f906e6ba5d5;l=22

[34] "Performance.cpp in in WebKit sourcecode," 2022, accessed: 2024-04-19. [Online]. Available: https://trac.webkit.org/browser/webkit/trunk/Source/WebCore/page/Performance.cpp

[35] "nsRFPService.cpp in mozilla sourcecode," 2024, accessed: 2024-04-19. [Online]. Available: https://searchfox.org/mozilla-central/source/toolkit/components/resistfingerprinting/nsRFPService.cpp#419

[36] C. Sahu, M. Banavar, and S. Schuckers, "A Novel Non-Linear Transformation Based Multi User Identification Algorithm for Fixed Text Keystroke Behavioral Dynamics," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 5, no. 2, pp. 277–287, Apr. 2023.

[37] V. Medvedev, A. Budžys, and O. Kurasova, "Enhancing Keystroke Biometric Authentication Using Deep Learning Techniques," in *2023 18th Iberian Conference on Information Systems and Technologies (CISTI)*, Jun. 2023, pp. 1–6.

[38] A. Thakare, S. Gondane, N. Prasad, and S. Chigale, "A Machine Learning-Based Approach to Password Authentication Using Keystroke Biometrics," in *Machine Learning, Deep Learning and Computational Intelligence for Wireless Communication*, E. S. Gopi, Ed. Singapore: Springer, 2021, pp. 395–406.

[39] S. Srivastava and P. S. Sudhish, "Continuous multi-biometric user authentication fusion of face recognition and keystoke dynamics," in *2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, Dec. 2016, pp. 1–7.

[40] S. Bleha, C. Slivinsky, and B. Hussien, "Computer-access security systems using keystroke dynamics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 12, pp. 1217–1222, Dec. 1990.

[41] R. Duda, P. Hart, and D. G.Stork, "Pattern Classification," in *Wiley Interscience*, Jan. 2001.

[42] L. Araujo, L. Sucupira, M. Lizarraga, L. Ling, and J. Yabu-Uti, "User authentication through typing biometrics features," *IEEE Transactions on Signal Processing*, vol. 53, no. 2, pp. 851–855, Feb. 2005.

[43] J. Huang, D. Hou, S. Schuckers, T. Law, and A. Sherwin, "Benchmarking keystroke authentication algorithms," in *2017 IEEE Workshop on Information Forensics and Security (WIFS)*, Dec. 2017, pp. 1–6, iSSN: 2157-4774.

[44] C. Murphy, J. Huang, D. Hou, and S. Schuckers, "Shared dataset on natural human-computer interaction to support continuous authentication research," in *2017 IEEE International Joint Conference on Biometrics (IJCB)*, Oct. 2017, pp. 525–530.

[45] N. Gonzalez, E. P. Calot, and J. S. Ierache, "A Replication of Two Free Text Keystroke Dynamics Experiments under Harsher Conditions," in *2016 International Conference of the Biometrics Special Interest Group (BIOSIG)*, Sep. 2016, pp. 1–6.

[46] N. Whiskerd, N. Körtge, K. Jürgens, K. Lamshöft, S. Ezennaya-Gomez, C. Vielhauer, J. Dittmann, and M. Hildebrandt, "Keystroke biometrics in the encrypted domain: a first study on search suggestion functions of web search engines," *EURASIP Journal on Information Security*, vol. 2020, no. 1, p. 2, Dec. 2020.

[47] C. M. Tey, P. Gupta, and D. Gao, "I can be you: Questioning the use of keystroke dynamics as biometrics," in *Network and Distributed System Security Symposium*, 2013.

[48] J. Leinonen, P. Ihantola, and A. Hellas, "Preventing Keystroke Based Identification in Open Data Sets," in *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*, ser. L@S '17. New York, NY, USA: Association for Computing Machinery, Apr. 2017, pp. 101–109.

[49] J. V. Monaco and C. C. Tappert, "Obfuscating Keystroke Time Intervals to Avoid Identification and Impersonation," Jan. 2017. [Online]. Available: http://arxiv.org/abs/1609.07612

APPENDIX

Tables V and VI, as well as the tools and scripts used for conducting the experiments are available at https://github.com/das-group/Robustness-of-Keystroke-Biometrics.