Whiskerd *et al. EURASIP Journal on Information Security* (2020) 2020:2
https://doi.org/10.1186/s13635-020-0100-8

EURASIP Journal on Information Security

RESEARCH Open Access

Keystroke biometrics in the encrypted domain: a first study on search suggestion functions of web search engines

Nicholas Whiskerd[1*], Nicklas Körtge[1], Kris Jürgens[1], Kevin Lamshöft[1], Salatiel Ezennaya-Gomez[1], Claus Vielhauer[1,2], Jana Dittmann[1] and Mario Hildebrandt[1]

Abstract

A feature of search engines is prediction and suggestion to complete or extend input query phrases, i.e. search suggestion functions (SSF). Given the immediate temporal nature of this functionality, alongside the character submitted to trigger each suggestion, adequate data is provided to derive keystroke features. The potential of such biometric features to be used in identification and tracking poses risks to user privacy.

For our initial experiment, we evaluate SSF traffic with different browsers and search engines on a Linux PC and an Android mobile phone. The keystroke network traffic is captured and decrypted using mitmproxy to verify if expected keystroke information is contained, which we call quality assurance (QA). In our second experiment, we present first results for identification of five subjects searching for up to three different phrases on both PC and phone using naive Bayesian and nearest neighbour classifiers. The third experiment investigates potential for identification and verification by an external observer based purely on the encrypted traffic, thus without QA, using the Euclidean distance. Here, ten subjects search for two phrases across several sessions on a Linux virtual machine, and statistical features are derived for classification. All three test cases show positive tendencies towards the feasibility of distinguishing users within a small group. The results yield lowest equal error rates of 5.11% for the single PC and 11.37% for the mobile device with QA and 23.61% for various PCs without QA. These first tendencies motivate further research in feature analysis of encrypted network traffic and prevention approaches to ensure protection and privacy.

Keywords: Keystroke dynamics, Biometrics, Search engines, Encrypted domain

1 Motivation

So-called search suggestion functions (SSF) are popular means to support users in typing search terms in text fields of search engines. These SSF are technically implemented by incremental, character-by-character server-side queries, where key-by-key information of the typing process is transferred to the servers, which then return proposed search term lists to the web browsers for display and potential selection by the users. Search engines use the Transport Layer Security (TLS) protocol for protection of the communication (authentication and encryption) between client and server. As keystrokes are input, the search engine creates encrypted data packets and forwards them to the search provider via the TCP/IP network. Typically, search suggestions are provided as the queries are typed, i.e. immediately after each keystroke. Exactly what data is shared with the search provider and how frequently packets are exchanged is not transparent. Therefore, the level of data disclosure involved in interacting with search engines is likely not something a typical user is aware of and consent is unclear.

Keystroke patterns, or more precisely, temporal features of key presses during the user's typing of text sequences, are commonly known to be potentially adequate for biometric user recognition. In this domain, biometric keystroke analysis, the dynamics of the typing are utilised to construct typical user patterns and to use those for biometric identification or verification. This can be achieved by means which is either text-dependent

*Correspondence: nicholas.whiskerd@ovgu.de
[1]Multimedia and Security Lab (AMSL), Otto-von-Guericke-University, Magdeburg, Germany
Full list of author information is available at the end of the article

or text-independent. If an instance of biometric recognition is possible, this would allow passive registration of a user during normal search behaviour by the search engine provider. Moreover, potentially a third party with access to the network data stream could classify similarly, even if only considering the subset of biometric information in the encrypted domain.

However, there are limiting factors on derivable keystroke information in the specific context of SSF. In the context of established keystroke dynamics, only a subset of conventionally usable keystroke events are available in the data captured from the network stream. When typing on a traditional hardware keyboard, key down events trigger input, and therefore, these are the only events which can be inferred from the respective captured network data. In this case, the distance measure from which biometric features can be derived is a "down-down" (DD) distance. On the other hand, conventional software keyboards such as those commonly used on modern mobile devices will only input characters on key up events. Thus, the distance measure for software keyboards is an "up-up" (UU) distance. Further to these limitations, dynamic network conditions of realistic environments will influence the time of capture for each the keystroke and risk damaging biometric usefulness or reliability. These factors are presented in Fig. 1 which breaks down the keystrokes and respective packets for the incrementally typed word "weather" on a hardware keyboard. The figure contrasts the conventional biometric perspective (i.e. involving dwell and flight times) with the limited derivable features from encrypted captured packets, which nonetheless may divulge significant biometric information.

Once a biometric keystroke pattern is built, the user could be later re-identified in further typing in the same applications (search engines) or within other applications and platforms. Even if the visit is anonymous, the keystroke pattern allows potential correlations to be made between visits and also across other applications, with further potential to divulge true identity. While discrimination in large populations may be difficult, the application within smaller domains is enough of a risk for identification, e.g. shared devices within families or school computer labs. Whether this data is processed (and, if it is, how) may not be determinable from the application perspective, but from the network perspective, a data collection is very possible. There are considerable legal requirements, e.g. GDPR [1], to abide by given the tracking and misuse potential.

The observation of the key-by-key transmission of user data to servers, in combination with the known potentials for keystroke dynamics, poses risks of unwanted user identification/tracking. As a prospective violation of user privacy, this motivates the work presented in this article. Thus, the following main contributions are made, which appear to be closely within the scope of this special issue "Biometric Authentication on Mobile Devices":

- Thorough *theoretical and experimental analysis* of potential flows of *biometrically relevant keystroke data* by SSF for different browsers, computing platforms, keyboard concepts and operating systems.
- Constitution of the relevance of *leakage of biometric keystroke data* in SSF both on desktop computers, as

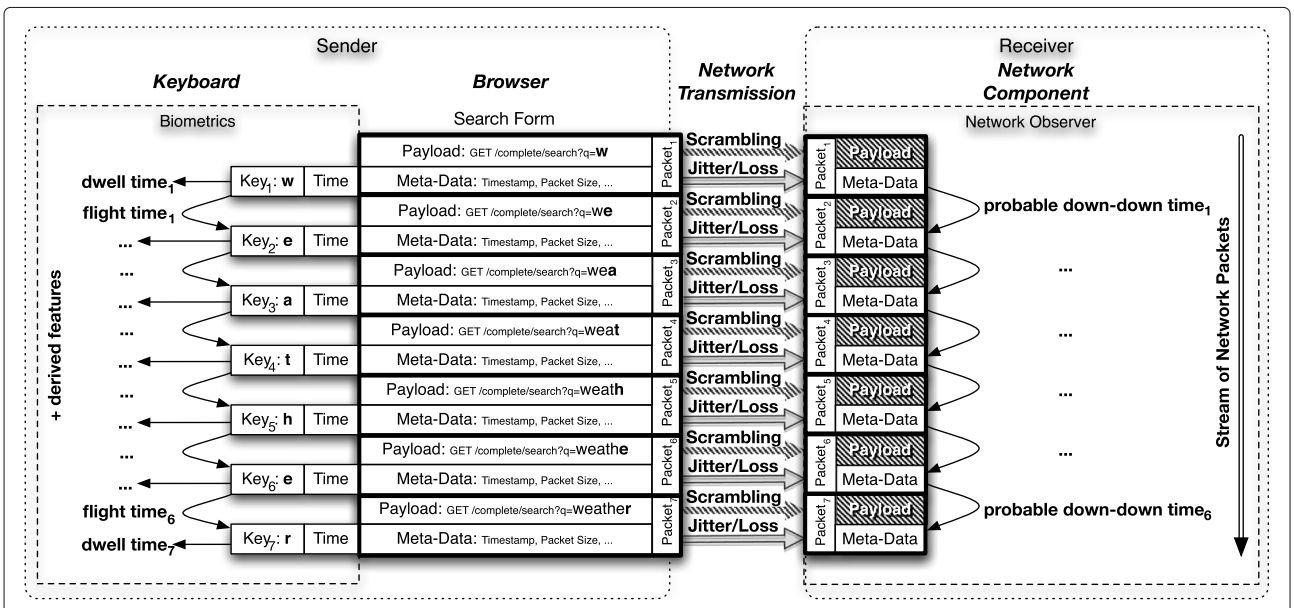

Fig. 1 Keystroke network transmission breakdown for word "weather" on hardware keyboard

well as *ubiquitous mobile computing platforms* such as smartphones and tablet devices.
- Identification and formulation of a *novel research challenge* involving biometrics on mobile devices, in regard to data security and *user privacy*.
- *Proof of concept* of the feasibility of biometric user identification based on timing features of metadata in encrypted network data packets, resulting from keystroke input to SSF.

With all these observations and privacy implications as incentive, the investigation in this article is focused towards two research questions as goals:

- Q_1: General analysis of the *relation between encrypted network packets* generated by SSF and the *temporal keystroke sequences*: how frequently are TLS protocol packets sent during typing of a search query and can these packets be assigned to the temporal relations between single, individual keystrokes?
- Q_2: Analysis of the *potential of biometric user* identification *by metadata in encrypted network data packets* generated by SSF: are there biometric features which can be derived from the TLS-encrypted packets based on the typing (keystroke dynamics) and, if so, how effective is this limited feature space for biometric recognition within the encrypted domain?

In approaching these research questions, three test goals are established which are formally defined in Section 3.2. The first phase of investigation (T_1) involves an overall assessment of behavioural tendency (outlined in Q_1) across four search engines S (Google S_1 and three others with a focus on privacy: Qwant S_2, DuckDuckGo S_3 and Ecosia S_4), two browsers B (Chrome B_1 and Firefox B_2), and two operating systems (Linux L and Android A). This assessment is done with two different input methods K: a hardware keyboard on Linux K_{HW} and a software keyboard on Android K_{SW}. A second phase of experimentation (T_2) involves a small data collection from five subjects $N_1 - N_5$ on both K_{HW} and K_{SW} with a focus on narrower selected test cases for one selected search engine and browser, which is chosen based upon the first phase findings. The decrypted network traffic captured with these criteria is used to create a proof of concept and to assess the impact of differing lengths of the search query phrase. The third and final experiment (T_3) involves ten anonymous subjects in more realistic and varied network conditions. For T_3, an open call for contributions was created in our laboratory with the option to upload the captured samples anonymously. The PCs used by participants differed in hardware and respective keyboards K_{HW}, but the experiment ensured the use of a common Linux virtual machine. From this setup, the encrypted data capture is focused upon in analysis for identification and verification potential.

The further article is structured as follows: Section 2 covers related work and presents conventional features we base our analysis upon; Section 3 details the technical specifications of our experimental setup, goals, and assumptions and limitations; Section 4 explains the methods we use in pre-processing and evaluating our captured data; Section 5 presents the full data collection specifics along with our results presentation and discussion; and Section 6 presents our conclusions.

2 Related work

History of keystroke dynamics dates back to early days of wireless radio, in which people transmitting Morse code messages could learn to identify each other by the way a person manually input the code, i.e. their unique "fist" [2], notably useful in verifying the integrity of military intelligence communications.

In recognition from typing, preliminary efforts established potential of keystroke dynamics to authenticate a user's access to a computer. Works [3] considered the moment each key is struck (key down) and the resultant distances between. Through the application of statistical models, the authors were able to present clearly distinct differences of different typists on a small population. These early principles were built upon in the work [4] where the authors consider lengths of time keys remain depressed as further features, along with research into "Free-Style" text (*free-text*) rather than only structured input (*fixed-text*).

Presently, the standard data that can be captured from hardware keyboards on modern computers are the times when each specific keys are pressed and released, i.e. the key down and key up events. These can easily be captured by specialised software designed to record keystrokes. Exactly how these key events are used to build features in keystroke analysis varies but are in principle built from temporal distances between these events. A subset of the following features is typically used [5], denoted as combinations of down (D) and up (U) events to represent the time differences between when:

- DU1: a key is pressed and the same one is released (dwell time)
- DU2: a key is pressed and the next key is released
- UD: a key is released and the next is pressed (flight time)
- DD: a key is pressed and the next key is pressed
- UU: a key is released and the next key is released

Based upon the choice of features, various machine learning techniques are often applied to classify the data [5] such as Bayesian classifiers.

Much of the study historically has been on hardware keyboards. Today, software keyboards are vastly popular on mobile devices, and this has opened up possibilities for further features to be derived from input and interaction, that is, beyond only when the keys are pressed and released [6]. More broadly, further information can be included for classification on both software and hardware keyboards, as there exists potential for combination with other biometric features or behaviours, e.g. patterns and habits in performing searches.

With regard to biometric recognition in the encrypted domain, a great variety of methods have been proposed already to protect biometric data during processing and communication to ensure privacy and confidentiality for the users [7]. These concepts typically involve some kind of cryptography and related key assignment to the entities involved in protocols, limiting access to biometric data to only those legitimate entities.

Works on biometric recognition based on entirely encrypted data, without any access to keys or plain unencrypted data, seem to be limited to date. However, some examples [8] demonstrate classification capability even with only limited features derived from network packets identified to contain keystrokes.

3 Experimental setup

To the best of the authors efforts, no earlier comparable publications on keystroke analysis based on network packets could be identified, and consequently, no previous test goals, protocols, or data can be referred to. Therefore, in this work, we propose a novel scenario of experimental analysis in applying keystroke biometrics to SSF and also present our own test data.

Our tests must therefore achieve an assessment of general tendencies in different test environments: how we observe differences in the keystroke-capturing behaviour exhibited by these different environments, based on the network packets we can capture and their contents. After successfully capturing data within the established conditions, we evaluate the biometric usefulness of the data.

We first present in Section 3.1 "Test setup" an initial overview of all components in our test environments and further specify in detail. The following Section 3.2 "Test goals" describes our intentions for use of the overall setup. Under Section 3.3 "Assumptions and limitations", we then present the constraints we place on our setup, in consideration of how this will affect result interpretation.

3.1 Test setup

The experiments T_1 and T_2 are performed on decrypted network traffic captured with a man-in-the-middle proxy (mitmproxy [9]). The experiment T_3 is performed on the foundation of the encrypted network traffic. The two different test setups are described in the following subsections.

3.1.1 *Decrypted network capture involving man-in-the-middle (T_1 and T_2)*

The overall setup for our investigation includes the known web-based client-server architecture enhanced with mitmproxy to capture network data.

This mitmproxy is the intermediary to the network connection of each of our test devices as shown in Fig. 2. By emulating the role of the search engine provider, mitmproxy receives the same data from the clients as the genuine provider would. Moreover, using mitmproxy enables the possibility to decrypt the network flow between the client and the proxy, which is useful for assuring that extracted timestamps from packets correlate to actual key events. We refer to this process of verifying expected packet contents by decryption as quality assurance (QA).

To collect the data, we use a setup consisting of a MacBook Pro 2017 running MacOS Mojave in our tests. This laptop uses a USB wireless dongle (Anadol AWL150 Micro with chipset Reling RT5370) routed to the virtual machine (Virtual Box 6.0.6 r130049) running on the laptop to host a WiFi network for our clients to connect to. The virtual machine itself makes use of Privacy International's data interception environment (February 2019) which is a Debian-based distribution. To inspect the traffic, mitmproxy runs (in "Regular" mode) on the virtual machine. Table 1 details the information of the client devices we use to connect to our proxy.

For cross-referencing, the behaviour of sending packets for each keystroke is tested in both browsers Chrome B_1 and Firefox B_2 on four different search engines: Google S_1, Qwant S_2, DuckDuckGo S_3 and Ecosia S_4. The test subjects input the data into search fields on each search engine website.

The network traffic from the client is proxied by mitmproxy to the search engine website. The communication between the client and mitmproxy is TLS-encrypted with a self-signed certificate and, therefore, can be decrypted for QA during the assessment of our research questions. As the timestamps of the client requests are non-encrypted metadata, decryption for the biometric feature extraction is not necessarily required. However, by doing so in this whitebox scenario, we are able to ensure that the extracted timestamps correlate correctly to the corresponding key-press event and opens up the possibility to detect packet loss or incomplete network captures. This possibility is especially useful for answering research question Q_1. In order to perform decryption, the certificate has to be installed on both client devices. The network packets are then TLS-encrypted (with the search engine certificates) and forwarded to the search engine by mitmproxy. Filtering the network flow for outgoing

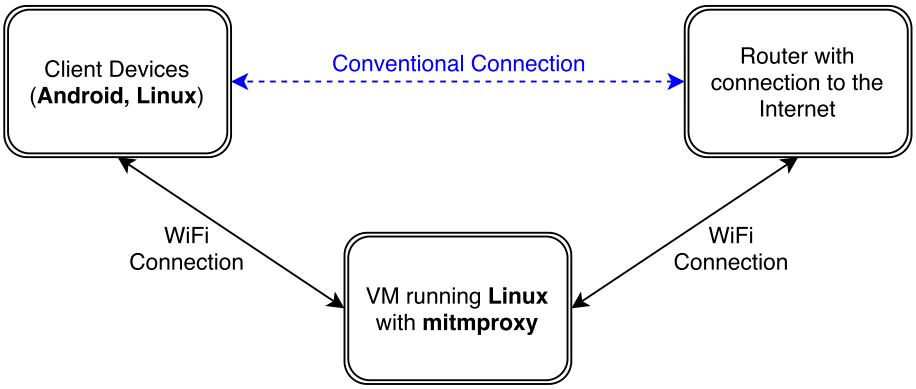

Fig. 2 T_1 and T_2 test setup

traffic and limiting to specific IP addresses of the search engines would narrow it down to only relevant packets, e.g. when working only on the encrypted stream without the decryption done by mitmproxy (such filtering is applied in the T_3 experiment).

In addition to our data collection from the network packets, logging keys locally with comparable timestamps allows examination of the delay. On the Linux hardware keyboard K_{HW}, we face technical limitations with no available functional keylogger which could record timestamps in milliseconds or better. However, for the software keyboard K_{SW}, we are able to make additions to the free software keyboard used (Simple Keyboard 3.71) to allow local keylogging of inputs to the nearest millisecond, comparable with the times extracted from the respective network packets. These alterations had no effect on the keyboard interface and user functionality.

3.1.2 Encrypted network capture (T_3)

Without access to the session keys an external observer will be limited to the analysis of encrypted TLS traffic directed to and coming from the search engine. Such an observer could gain access to the network communication at any node within the routing path between the user and the search engine provider.

The overall setup for our investigation in T_3 consists of a virtual appliance for Virtual Box running Ubuntu 18.04 with Firefox 70.0.1 (B_2) as shown in Table 1. The network traffic to and from this virtual machine V is automatically captured using Wireshark. Our overall intention in using a virtual appliance is keeping the software component constant during the experiment. Moreover, the captured network traffic will be limited to the virtual machine. However, in contrast to the experiments T_1 and T_2, this virtual machine is deployed on different computers, equipped with various hardware keyboards K_{HW}. The experiment is limited to Google S_1 as the search engine.

In order to gather the keystroke data for T_3, we developed a test protocol consisting of four total sessions. In each session, two search strings are typed five times alternating between the two to minimise learning effects during the session. In addition to that, there is a required break of at least 2 h between sessions of the same subject.

3.2 Test goals

Based on our motivation and research questions, we establish the following specific test goals T (specific fixed search queries are described in Section 4):

T_1 – To find tendencies in data disclosed to search engines under various conditions. The effects of the following factors are evaluated:

> $T_{1.1}$ – Four different search engines, Google S_1 and three with a focus on privacy: Qwant S_2, DuckDuckGo S_3 and Ecosia S_4.

Table 1 Client devices used for data collection: physical devices for both T_1 and T_2, whereas virtual setup for T_3

Client devices	Android (*A*)	Linux PC (*L*)	Linux VM (*V*)
Model	Samsung Galaxy S4 GT-19505	Clevo M570TU	Various
Operating system	LineageOS 14.1	Ubuntu 18.04	Ubuntu 18.04
Chrome version (B_1)	75.0.3770.67	74.0.3729	-
Firefox version (B_2)	67.0	67.0.04	70.0.1
Keyboard	Simple Keyboard 3.71 (K_{SW})	Standard inbuilt keyboard (K_{HW})	Various hardware keyboards (K_{HW})

$T_{1.2}$ – Two different browsers: Chrome B_1 and Firefox B_2.
$T_{1.3}$ – Hardware keyboard K_{HW} on Linux OS and software keyboard K_{SW} on Android.
$T_{1.4}$ – Phrase input behaviour while SSF are or are not correct, no predictions can be made or suggestions are presented, and when the same phrase is deleted and re-entered.

T_2 – To determine if a subject can be identified in a small population based upon data received by a search engine provider. Due to the findings from the T_1 investigation—presented in Section 5—T_2 uses the conditions of search engine S_1 and browser B_1 for both K_{HW} and K_{SW}.

$T_{2.1}$ – Identification performance differences between keyboards on K_{HW} and K_{SW}.
$T_{2.2}$ – Identification performance differences of three search query phrases differing in length and commonality.

T_3 – To determine if a subject can be identified in a small population based upon encrypted data directed to a search engine provider but received by an external observer. T_3 uses the conditions of the same search engine S_1 but a different browser B_2 in comparison with T_2 for K_{HW} in V.

$T_{3.1}$ – Evaluation of the performance in a verification mode for V with K_{HW} using two different search strings.
$T_{3.2}$ – Identification performance differences between different search string lengths.

3.3 Assumptions and limitations

Our assumptions mostly relate to our test setup, especially to the network infrastructure. A relevant overview for the T_1 and T_2 setup is shown in Fig. 3. Our setup limits us to inspect on the (TLS-encrypted) network layer. Furthermore, we consider an optimal setting and perform no error handling, with regard to the following aspects:

1. We consider only a simple network without influence from high or low levels of network traffic, switched networks, or variable travel distances. Therefore, we do not experience fluctuating travel times for packets between client devices and the proxy.
2. We make the assumption that every packet is delivered on the first attempt. When collecting data for T_2, we only include phrases with both the first and last characters submitted successfully, such that we have our defined start and end points, and the full phrase for analysis. In some cases, no new GET requests are captured by the proxy. This could either be due to packet loss, the behaviour of the search engine provider, or other reasons unknown. We refer to this occurrence as *packet omission*. If this occurs for the first or last character in a test phrase, all corresponding packets to this particular phrase instance are removed from the data collection. For the T_3 experiment, where we do not have the possibility to decrypt the captured keystrokes and, therefore, cannot verify completeness of the input phrases (QA), this pre-processing obviously is not possible. Thus, in T_3, we consider each (encrypted) keystroke to be as expected.
3. In testing with subjects, an assumption is made that they input their search terms correctly without any typing errors for each phrase. Due to increased complexity in considering mistyped phrases, we limit ourselves to analysis of perfectly input samples, all inputs with errors are discarded. The captured data is not changed in any other way.

In addition, for T_3, a limitation of the experiment could be the uncontrolled usage of various computer hardware and Internet connections. This is a result of the increased number of test subjects for T_3.

4 Evaluation procedure

The sets C_{HW} and C_{SW} contain all possible keyboard inputs on K_{HW} and K_{SW}, respectively. We define a set of all keyboard inputs as $C_T \subseteq C_{HW} = C_{SW}$. Moreover, a set of possible phrases P, generated from set of keyboard inputs, is defined as $P = \{p \mid p = c_1 c_2 \dots c_n \mid c_m \in C_T \wedge n, m \in \mathbb{N}\}$. Our set of phrases for T_2 tests is $P_{T_{2A}} = \{p_1, p_2, p_3\} \subset P$ for B_{1A} (Android) and $P_{T_{2L}} = \{p_1, p_3\} \subset P$ for B_{1L} (Linux). For T_3, the set used is $P_{T_3} = \{p_1, p_2\} \subset P$ for B_{2V}. The corresponding phrases are:

- p_1 = "weather"
- p_2 = "security and biometrics"
- p_3 = "department of computer science"

The targeted sample collection totals for these phrases are presented in Table 2.

Additionally for the initial T_1 exploratory tests, the phrase p_0 "weather oldtown" is used to cover search inputs in line with $T_{1.4}$. The phrase p_0 is always correctly predicted and suggested throughout the first word, but when entering the second word, never correctly predicted as the full phrase through SSF (on any search engine tested). This allows comparison of any keystroke sending behaviour both while predictions are and are not correct.

For each keyboard input, we measure the corresponding timestamp using our test setup (see Section 3.3). In our experiments across both K_{HW} and K_{SW}, it is important to make a distinction in the features captured given the different default behaviours of input to the search field:

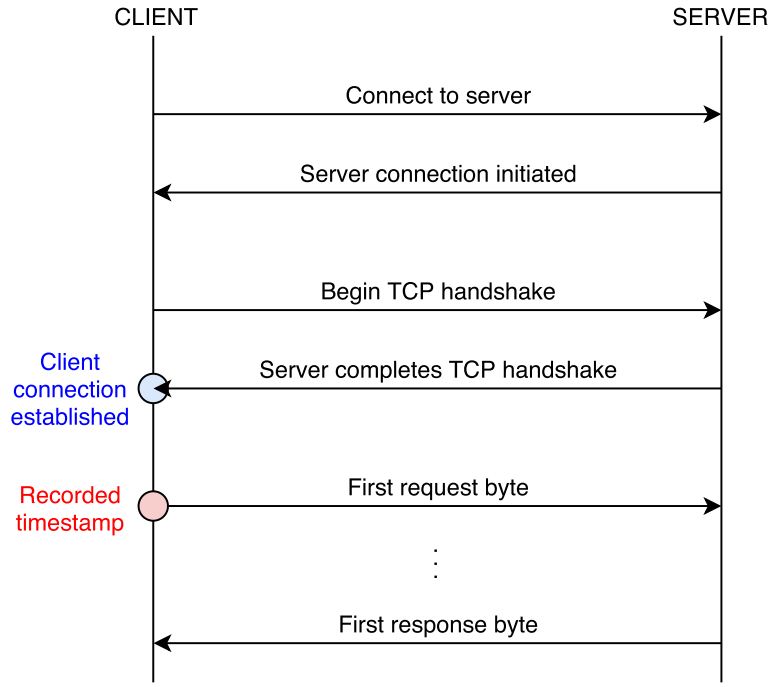

Fig. 3 T_1 and T_2 captured network traffic

- K_{HW}: Keys are input immediately on each key-down instance.
- K_{SW}: Due to the typical behaviour of K_{SW}, different characters can be selected by holding a key, and selection is only confirmed and input on release of the key, i.e. each key-up instance.

Assuming consistent delay, the time difference between K_{HW} keystrokes captured is DD and the K_{SW} distance is UU (as introduced in Section 1). Both of these have been shown to be viable features of comparable performance [10], therefore, we use these temporal distances in evaluation for both T_2 and T_3 (though the distinction between the types of events limits direct comparison across the two environments).

Due to the conceptional difference in the consideration of decrypted network packets for QA in experiments involving T_1 and T_2, and encrypted packets in T_3, we suggest two complementary procedures using temporal distances as features. In all circumstances, the timestamps are based on the capture times of the network packets (observer time). In T_1 and T_2, the timestamps originate from mitmproxy, whereas they are captured within the virtual appliance in T_3. Section 4.1 details evaluation procedures for T_1 and T_2, whereas Section 4.2 presents details on the approach for T_3.

4.1 Decrypted keystrokes for QA (T_1 and T_2)

The experiments T_1 and T_2 both consider temporal distances extracted from decrypted network packets. Pre-processing steps are similar for both T_1 and T_2, but the processes involving feature extraction and subsequent classification are exclusively in line with test goal T_2 in this section. An example for extracted features is shown in Fig. 4.

4.1.1 *Pre-processing*

According to T_2 test setup, data is captured and recorded from both B_{1L} and B_{1A} for all phrases P_T. A Python script extracts the timestamps of the inputs from the file (.cap) generated by mitmproxy for each keystroke. As shown in Fig. 3 the timestamp of the first packet sent from the client to the search engine provider, containing the request string (in mitmproxy called first request byte), is used.

Table 2 The targeted sample totals for collection in T_2 and T_3 experiments

Test goal	Subjects	Browser – search engine	Phrase	Samples per subject	Total samples
T_2	5	$B_{1A} - S_1$	p_1	10	50
			p_2	10	50
			p_3	10	50
		$^*B_{1L} - S_1$	p_1	40	200
			p_3	10	50
T_3	10	$B_{2V} - S_1$	p_1	20	200
			p_2	20	200

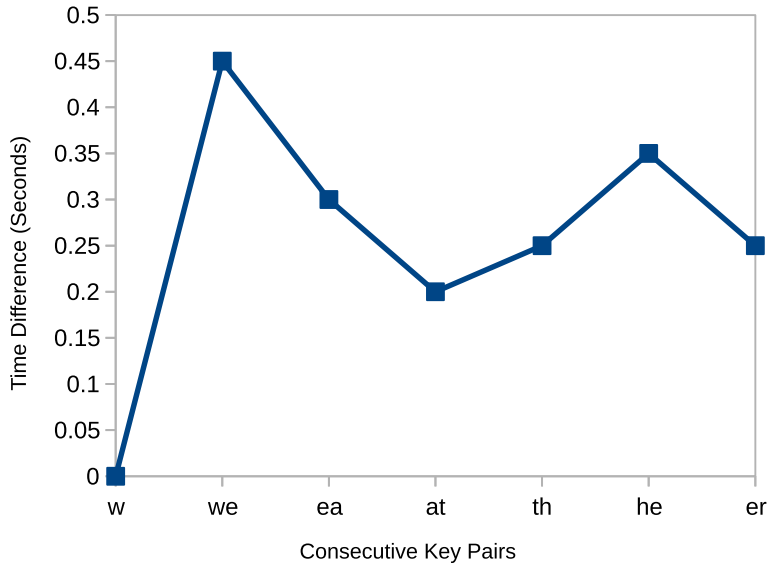

Fig. 4 Basic T_2 example (artificial data) for the phrase p_1 with time differences between each consecutive letter as features

Even though no decryption is needed as the used timestamps are metadata of when packets arrived at the proxy, we use the decrypted stream between client and proxy only as QA in our investigation to determine changes in characters or if packets were omitted.

The file structure contains packets in sequence, from which we examine the GET requests generated by a change made in the input search query. The irrelevant GET requests, e.g. for image data, are ignored. The sequential order we capture is not reliably the actual order in which the inputs were typed, but we sort our extracted keystrokes by the extracted timestamp of the first request byte of each packet. Within each such packet, the current query phrase string is sent as part of the request, e.g. p_3 can be seen as "q=department%20of%20computer%20science" after packet decryption with mitmproxy. In comparison with the previous string, it is viable to determine whether it is a newly input character, character deletion, or, in some cases, whether packet omission has occurred.

In cases of intermediary packet omission, there are steps taken either to remove or salvage samples. As mentioned in Section 3.3, if the first or last character of a phrase is not captured we do not consider it to be a usable sample and therefore immediately discard it. However, if any other timestamps within the sample phrase are omitted as a result of measurement or network errors, we estimate the missing data by one-dimensional linear interpolation. This divides the time difference between the captured characters spanning the window containing omission(s) equally amongst the missed characters. For example, if p_1 ("weather") is missing characters "t" and "h", the time difference from "a" to (the second) "e" will be divided by three and produce equal estimated "at", "th" and "he" distances. This example is visually presented in Fig. 5. This method of interpolation ensures we have a full set of features in the sample for analysis. Nonetheless, in the following Section 5.2, for T_2, we consider our results when using purely the complete samples as well as with those interpolated.

Due to variability in how many samples were gathered from each subject (explained in Section 5.1.2), and in the interests of avoiding distortion of results, the amount of samples across subjects for each class are balanced. With this approach, we can present and compare results when classification uses either balanced or unbalanced classes. When considering only our complete samples in the experiment, excess samples are removed from the set in overrepresented classes, specifically the latest captured ones (in consideration of reducing the effects of learning). In the case of our interpolated samples, we first prioritise removing the samples with the most interpolations for the sake of sample quality, then similarly further remove any by latest captures if necessary until all classes are balanced. Table 3 shows the amount of instances of each class, which is the same total for each subject, after balancing. From the table, it is evidenced that tests with at least one subject only captured two and one samples for p_1 and p_2 respectively. This motivates our interpolation method to make more samples, and therefore these classes, usable in subsequent classification.

The result of these steps is a set of files (.csv) with extracted timestamps and corresponding query phrases,

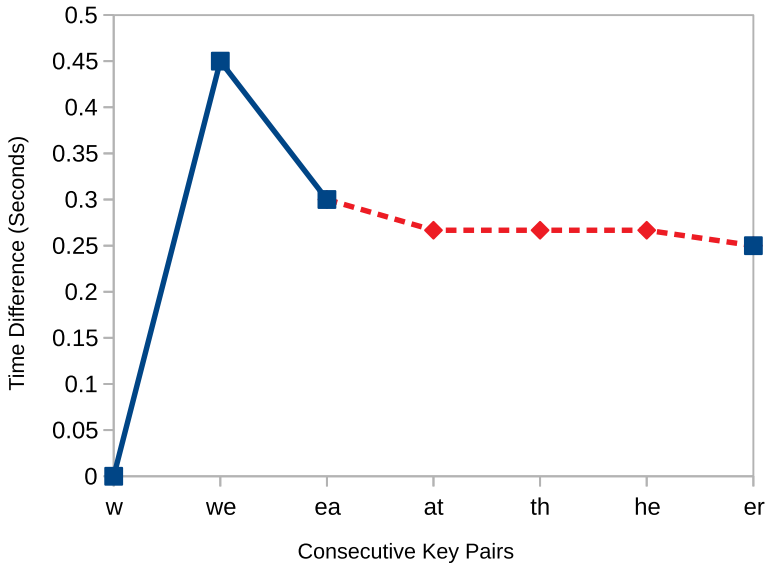

Fig. 5 Incompletely captured T_2 example (artificial data) with raw timestamps for "t" and "h" omitted. The time difference values calculated following the interpolation process are shown through the dashed line

sorted by timestamp. The data is pre-processed in all the possible permutations of including interpolations and/or balancing classes.

4.1.2 Feature extraction

In order to evaluate our data, we define empirical values that we can use to classify the data. Overall, for each phrase used in our tests, we have an initial feature set comprised of all the total distances between each neighbouring character in the phrase. Additionally, the average is calculated and appended to each sample as an additional feature. Therefore, for each of our phrases, we have feature totals: 7, 23 and 30.

Our evaluation data M is defined as pairs of sequences of characters of the test phrases in P_T and the corresponding timestamps of each sequence, that is $M = \{\langle c_1, t_1\rangle, \langle c_1c_2, t_2\rangle, ..., \langle c_1c_2...c_n, t_m\rangle \mid n, m \in \mathbb{N}\}$.

We calculate the time difference between two consecutive keyboard inputs c_i and c_{i+1} simply as:

$$d_{i,i+1} = t_{i+1} - t_i \mid i \in \mathbb{N} \tag{1}$$

When applied to all characters of a phrase $p \in P_T$, we obtain a set $D_p = \{d_{1,2}, ..., d_{n-1,n} \mid n = |p|\}$ of time differences for all consecutive key pairs.

We calculate the average (mean) for each phrase as an additional feature. The average over all time differences between two consecutive keyboard inputs for one phrase can be calculated by:

$$D_p = \frac{1}{n-1}\sum_{i=1}^{n-1} d_{i,i+1} \tag{2}$$

4.1.3 Classification

For every phrase p entered by a test subject, we have a set $F_p = \{d_{1,2}, ..., d_{n-1,n}, D_p\}$ of all features of that particular phrase. With samples of the same phrase, we can try to classify each subject by some classification algorithms. We choose Gaussian naive Bayesian (NB) and nearest neighbour by Euclidean distance (NN) as classifiers specifically, due to their effective performance with small data sets.

4.2 Encrypted keystrokes without QA (T_3)

The experiment T_3 utilises temporal distance features extracted from the encrypted data stream. As the data is encrypted, this means definitive QA of packet content for correct key information is not possible. The classification is performed on the foundation of the Euclidean distance between the feature vector and a subject's template.

Table 3 From T_2 tests, amount of samples remaining after balancing, per subject (e.g. 41 × 5 = 205 total for B_{1L} - S_1 - p_1)

Browser – search engine	Phrase	Only complete samples	Including interpolated samples
$B_{1A} - S_1$	p_1	2	8
	p_2	1	8
	p_3	6	9
$B_{1L} - S_1$	p_1	41	-
	p_3	9*	-

*Only nine complete samples were collected from one subject due to human error in the capture process setup

4.2.1 Pre-processing

The pre-processing of the encrypted data captures consists of the following steps:

1. Determination of the IP addresses of the search engine by analysing DNS responses from the name server.
2. Extraction of candidate type sessions based on the determined target IP addresses.
3. Selection of the pre-sessions based on the session length.

Firstly, the capture files are read using Scapy and analysed towards the DNS responses for the specified search engine domain. In our case, the search engine is limited to S_1 (in top-level domain .com). The DNS response contains one or multiple IP addresses which are contacted in order to perform the search. Each IP from the DNS response is stored in a set to compare with the destination IP of each network packet.

During the second step, sessions are extracted based on their metadata: packet length between 208 and 300 bytes and monotonic increase of the packet size in comparison with the previous packet to the same destination IP, which is equivalent to two consecutive key-presses resulting in key down-down times. Due to the compression and encryption of the requests, we can assume that the next network packet including one additional character will be either as large as its predecessor or one byte longer. If the packet size is lower or significantly larger, the session will be terminated. Furthermore, sessions with less than five packets will be discarded. If the inter-packet-time exceeds 5000 ms, the session will be terminated as well.

For the last step, the candidate sessions are manually selected based on the session length:

- Between 5 and 9 packets: p_1 ("weather")
- Between 21 and 24 packets: p_2 ("security and biometrics")

The intervals are chosen because packets might be omitted in the capture file. Furthermore, additional packets sent during the typing session might sporadically interfere with the total number of detected packets. This selection process is a realistic consideration because an external observer would also have to deal with the same issues.

4.2.2 Feature extraction and selection

From the extracted sessions, statistical features are extracted based on the timestamps of two consecutive packets within the session. Overall, we differentiate between a set of length-independent features F_1 and a length-dependent feature in F_2 from an encrypted session E with $n \in \mathbb{N}$ packet timestamps $e_n \in E$:

1. $F_1\text{min} = \underset{e_n \in E, n>1}{\arg\min}(e_n - e_{n-1})$
2. $F_1\text{max} = \underset{e_n \in E, n>1}{\arg\max}(e_n - e_{n-1})$
3. $F_1\text{mean} = \bar{e} = \frac{1}{n-1}\sum_{i=2}^{n} e_i - e_{i-1}$
4. $F_1\text{variance} = \sigma^2 = \frac{1}{n-1}\sum_{i=2}^{n}((e_i - e_{i-1}) - \bar{e})^2$
5. $F_1\text{skewness} = \frac{1}{n-1}\sum_{i=2}^{n}(\frac{(e_i - e_{i-1}) - \bar{e}}{\sigma})^3$
6. $F_1\text{kurtosis} = \frac{1}{n-1}\sum_{i=2}^{n}(\frac{(e_i - e_{i-1}) - \bar{e}}{\sigma})^4$
7. $F_1\text{median} = \tilde{e}$
8. F_1regressslope : slope of the linear regression of inter-packet-times
9. $F_2\text{total} = e_n - e_1$

From these features, we calculate the templates for each test subject based on the first three of the four sessions. In particular, we create three templates—for p_1, for p_2, and for all samples from p_1 and p_2: p_1p_2. The templates are calculated by determining the mean of all training/enrolment sessions (15 out of 20 samples). In addition to that, we determine the intra-person-variance as a foundation for the feature selection.

The feature selection is performed empirically by comparing the equal error rates (EER) after selection of the subsets of features. In particular, a variance threshold of 0.5 for each feature, resulting in selection of the features F_1min, F_1mean, F_1median and F_1regressslope which yield the best performance based on our data set.

4.2.3 Classification

The matching is performed by a standard template matching approach using the Euclidean distance between the trained template and a feature vector. In addition, we have evaluated the Manhattan distance as well as the Canberra distance; however, both yielded lower detection performances. We evaluate the biometric system in the verification mode and in the identification mode up to a rank level of three.

5 Experiments

Firstly, in Section 5.1 "Data collection" we detail what data we actually captured through use of our test setup. This is followed by Section 5.2 "Results" in which our analysis is discussed and presented for each of our test goals in turn.

5.1 Data collection

5.1.1 T_1 collection

To address test goals $T_{1.1} - T_{1.4}$, the data was collected across the specified browsers B_{1L}, B_{2L}, B_{1A} and B_{2A}, on search engines $S_1 - S_4$ with the phrase p_0 as introduced in Section 4. Two subjects were used in completing the

inputs across two days. The website of each respective search engine was visited, the input was made into the search bar provided, and then it was submitted with the enter key. Subsequent searches and amendments to the term were made on the results page where a search field remained accessible.

To test any keystroke sending behaviour when the same data is re-entered, first "weather" is typed, deleted by backspace, and then re-entered exactly. The second "oldtown" input is entered and then a similar test is performed after completion of the full phrase: the phrase was partially deleted by the backspace key to leave only "weather", and then "oldtown" was re-entered. To test the behaviour while suggestions were no longer visibly generated, further non-specific words were input—chosen in direct contradiction to any further search suggestions—until they were no longer provided, then one word more was typed beyond that boundary.

5.1.2 T_2 collection

All data was collected on a single day for the K_{SW} and across two neighbouring days for the K_{HW}. The test subjects typed all instances of each phrase in one single session.

We used the three phrases defined in Section 4. The phrase p_1 was chosen specifically because it is one of the most commonly searched terms across all search engines and, therefore, serves as an indication of whether the typing behaviour can be used to track people. The other two terms p_2, p_3 were chosen as examples of longer, more uncommon terms as per $T_{2.2}$.

Each phrase was first typed five times to train the person on the unfamiliar keyboard and phrase. Then each person entered the phrase at least ten times correctly. Any phrases with mistakes were discarded at this stage as described in Section 3.3. Any method for clearing the text was permitted, such that the next sample could be input into a newly blank search field.

The subjects came from different language backgrounds, and while all use both kinds of keyboards daily, the keyboard layout usually used differed. The keyboard layout did not get swapped for any subject. The K_{HW} used was a German language layout keyboard; however, only phrase p_2 involved a key not identically placed to that of an English keyboard (Y/Z). The K_{SW} was used with a default English keyboard layout.

Every test for T_2 was conducted while using search engine S_1 on the Chrome B_1 browser on both devices. For the initial tests with the K_{SW}, at least ten samples were collected from each subject for each phrase: p_1, p_2 and p_3. In each case, the subject was asked to be certain if they had input the full amount of phrases correctly without typing errors; therefore, in some cases, more than ten samples were collected.

For the second round of tests involving K_{HW}, ten p_3 samples were collected in a similar manner. However, it was decided while testing to focus on at least 40 inputs for p_1 per person. This change was motivated by issues raised from the small sample sizes of initial results gathered from the K_{SW}, discussed in Section 5.2.

5.1.3 T_3 collection

The sessions for T_3 are recorded within a fixed software setup provided by the virtual machine V (using B_{2V} with S_1). The test subjects were instructed to type p_1 "weather" and p_2 "security and biometrics" five times within each session. Hereby, the keyboard layout should be ignored and all subjects should type with the layouts they are accustomed to. Since the actual content of the search string cannot be determined due to the TLS encryption, the impact of an incorrect search string is negligible. However, the test subjects are instructed to abort the current search if they recognise a mistyped character. In total, four sessions are recorded by each of the test participants yielding 20 samples for p_1 and p_2 per subject.

5.2 Results

5.2.1 T_1 presentation

In Table 4, the tendencies of sending keystrokes are presented for the scenarios outlined in $T_{1.1} - T_{1.3}$, denoted by symbols corresponding to disclosure levels of *full*, *irregular* or *none*, which we define as follows:

Table 4 Results for $T_{1.1} - T_{1.4}$

Browser	Search engine	During suggestions	During no suggestions	During re-entry
B_{1L}	S_1	●	●	●
	S_2	●	●	◐
	S_3	◐	◐	◐
	S_4	●	○	○
B_{2L}	S_1	●	●	●
	S_2	●	●	◐
	S_3	◐	◐	◐
	S_4	●	○	○
B_{1A}	S_1	●	●	●
	S_2	●	●	◐
	S_3	◐	◐	◐
	S_4	●	○	○
B_{2A}	S_1	●*	●	●
	S_2	●	●	◐
	S_3	◐	◐	◐
	S_4	●	○	○

● full, ◐ irregular, ○ none

*Unique behaviour: if the present phrase entered continues to match all (in this case five) displayed SSF, there are no further keystrokes sent until the phrase no longer matches SSF

- *Full*: Up to (but no more than) two consecutive keystrokes may not be sent in these conditions. We consider this a "full" tendency given the general behaviour to send every keystroke.
- *Irregular*: At least one occurrence is recorded where more than two keys in a row are omitted from phrases. Therefore, incomplete samples are regularly captured.
- *None*: No evidence is found of keystrokes being sent in packets.

The latter three headings of Table 4 address the $T_{1.4}$ process as detailed in Section 5.1.1; in summary, these are the situations during which:

- *Suggestions*: Entry while SSF are predicted and presented to the user.
- *No suggestions*: Entry as further text is entered such that SSF have ceased.
- *Re-entry*: The phrase entered (which SSF had been generated for) is deleted by backspace and then retyped exactly, intended to trigger the same suggestion.

5.2.2 T_1 discussion

Our exploratory data collection across these scenarios demonstrates reliable keystroke collection to some extent across three of the four search engines. With such data disclosure occurring, this suggests potential for the data to be used in recognition applications.

We note that regardless of the conditions in our tests, whenever a packet was successfully sent on keystroke input, the size difference was always the same for any individual keystroke. While from our results we observe that packets were not always sent due to packet omission, this irregularity was a behaviour observed to be independent of how fast or slow a user types.

One search engine behaviour shown in Table 4—which is not transparent to the user—is how keys are sent even when SSF are no longer presenting suggestions. This is true for S_1, S_2 and S_3, while S_4 does not send keystrokes after suggestions can no longer be predicted.

Another observation present in Table 4, is the re-sending of keystrokes when text is deleted and re-entered, therefore, triggering the same suggestions as before. This user action noticeably reduced the sent keystrokes for S_2 and S_3, with none sent for S_4, which simply reused the previously requested suggestions. S_1's behaviour was consistent in reliably sending regardless of circumstance.

To partially address the reason for differing behaviour between search engines on Linux and Android, it is to be noted that the "mobile" versions of websites differ from "desktop" counterparts. This difference is at the very least visible in presentation, but also may be responsible for different underlying behaviours.

Due to the highest reliability and consistency observed by B_1 and S_1 across both the Linux and Android operating systems, this was selected as the preferred environment for addressing T_2.

5.2.3 T_2 presentation

For classification of each set of samples, both the NB and NN classifiers are applied. Classification is performed using Python through use of the scikit-learn packages [11]. To split testing and training data, fivefold cross validation is used, and for each fold, the macro average of the EER is calculated. The EER presented in the Table 5 is the mean of these averages across all five folds for each phrase on each device. Two concatenation combinations of $p_1p_2p_3$ for K_{SW} and p_1p_3 for K_{HW} are also included.

Results for both unbalanced and balanced classes are shown. The results also show when samples with missing values are excluded or included with interpolations. For K_{SW} phrases p_1 and p_2, when missing values are excluded, the EER cannot be calculated as there are too few instances of some classes for fivefold cross validation. Therefore, interpolation is a prerequisite for evaluation by our classification process in these two cases.

5.2.4 T_2 discussion

In overall observation of the results of Table 5, we can see lower EER results from K_{HW} than K_{SW}. For example, considering our longest single phrase p_3 with balanced classes in the samples, for K_{SW}, we see EER of 18.04% and 22.28% for NB and NN respectively, whereas for K_{HW} EER 5.11% and 9.44% are the respective values. Similarly with our shortest phrase p_1, balanced class results for K_{SW} of 24.12% and 34.36% are markedly lower for K_{HW} at 12.97% and 16.83%.

One reason for lower EER with K_{HW} is likely the more reliable data capture of the environment. Despite ensuring correct input and using the browser and search engine combination deemed most reliable in T_1 for K_{SW}, some expected packets were omitted; therefore, we had incomplete samples. While still usable to an extent through interpolation, omission is information loss and naturally would be expected to affect performance. Furthermore, the familiarity with the K_{HW} environment would make inputs more consistent as all subjects have previously used a physical keyboard with a similar layout. The K_{SW} on the other hand is on a smaller device, which subjects may not be used to, hindering natural movements. Nonetheless, what likely had the most impact would be that subjects are instructed to type without K_{SW} predictions/suggestions and expected to make no mistakes. Given the prevalence of keyboards with auto-corrective systems, users would

Table 5 Results for $T_{2.1}$ and $T_{2.2}$: fivefold cross validation equal error rates (EER) for each phrase under Gaussian naive Bayes (NB) and Euclidean nearest neighbour (NN)

$B_i - S_i$	$p \in P_T$	Missing values	Balanced	NB (EER%)	NN (EER%)
$B_{1A} - S_1$	p_1	Excluded	✗	–	–
			✓	–	–
		Interpolated	✗	18.48	30.11
			✓	24.12	34.36
	p_2	Excluded	✗	–	–
			✓	–	–
		Interpolated	✗	19.81	26.06
			✓	12.18	25.36
	p_3	Excluded	✗	16.47	24.04
			✓	18.04	22.28
		Interpolated	✗	25.58	27.75
			✓	27.56	27.77
	$p_1p_2p_3$	Interpolated	✗	17.18	13.34
			✓	11.37	13.35
$B_{1L} - S_1$	p_1	N/A	✗	15.49	16.31
			✓	12.97	16.83
	p_3	N/A	✗	9.18	9.41
			✓	5.11	9.44
	p_1p_3	N/A	✗	11.17	12.29
			✓	9.73	15.60

Results are shown for when samples with missing values are excluded (not always possible if too few samples) or included with interpolations and for sample sets with either unbalanced or balanced classes

be accustomed to making frequent small mistakes, which contrasts with our assumption of a totally flawless input.

Concatenation of the three phrases showed clear improvement in EER for K_{SW}, with balanced classes the lowest K_{SW} EER value of 11.37% is achieved by NB with concatenation. The improvement is greater for NN with an EER value of 13.35% compared to the previous (balanced and interpolated) lowest of 25.36%.

The concatenation of p_1 and p_3 for K_{HW} also showed lower EER results when compared with p_1. However, EER values for p_3 alone are the lowest. Due to disparate sample totals for p_1 and p_3 (41 and 9 respectively in the case of balanced classes), this meant concatenation of the two is limited to producing nine samples. As only one concatenation configuration was trialled, this coincidentally may have included the least effectively distinct intra-class samples for p_1 to concatenate with p_3, increasing EER values by NB and NN, rather than decreasing EER values as desired.

To address the issue of missing values captured from the K_{SW}, we assess results both when only complete samples are considered, and when samples with interpolated values are also included. However, as only p_3 contained enough samples for each class for fivefold cross validation, we can only examine the differences for p_3. Interpolation here appears to increase the EER value, suggesting samples with interpolation are not significantly valuable and in some cases may produce samples without utility for classification. Introducing a maximum threshold for interpolations within a phrase may be a compromise to make more samples usable without risk of filling them with too much insufficiently meaningful data.

5.2.5 T_2 latency

To further assess the conditions of our T_2 results, we review our assumption on latency. If the packet goes through the Internet and to the search engine provider, in reality, due to variation in traffic or route distances, the travel time could be variable. Therefore, interpretation of our results assumes a stable travel time despite these potential conditions. However, beyond this assumption, due to our captured local keystrokes, some basic statistical analysis can be performed on the average delay in comparison with network results. Capturing p_2 on the K_{SW} is the scenario with the most omitted packets for one of the subjects—only one complete sample from N_1 without interpolation can be derived—so we use the p_2 collection as our most diverse example for latency comparison.

Table 6 shows the delay on our sessions for each subject for inputs of p_2. As client and server are running on unsynchronised clocks, we consider "delay" to be additional time beyond the minimum difference between

Table 6 Latency effects on T_2: p_2 keystrokes delay beyond minimum offset (ms)

Subjects	Max	Average
N_1	1239	601
N_2	88	56
N_3	78	51
N_4	697	267
N_5	251	202

timestamps as an offset for each subject, e.g. for N_1, the minimum difference was 23,683 ms, so we make the assumption that the clocks were 23,683 ms out of sync in p_2 collection for N_1, and look at average and maximum delays beyond that baseline offset. Table 7 summarises the absolute value differences in feature calculations; the features calculated based on local timestamps are compared with those calculated based solely on the network timestamps.

For phrase p_2, packet omission occurred for N_1, N_4 and N_5. Most frequently for N_1—with the largest delay in Table 6—and least frequently for N_5—with the smallest delay of those with omissions. In this case for N_2 and N_3, we saw no instances of packet omission, and these display the shortest delays of under 100 ms even at their maximum.

While this is limited data for comparison, the average difference between local and network features is under 100 ms for all subjects, suggesting that while adverse network conditions affect the delay and likely increase packet omission, even in poorer network conditions, the delay was consistent to such a degree that the features which are captured remain very close to the true locally captured values.

5.2.6 T_3 presentation

The results for the evaluation of verification performance $T_{3.1}$ with the ten test subjects are depicted in Fig. 6.

Table 8 summarises the results of the identification within the scope of $T_{3.2}$ on the encrypted data stream up to a rank level of three. An attempt to identify a subject is considered successful when the correct identity is contained within a ranked list. The rank of the list describes the number of its entries; for rank 1, the template with the minimum distance is the only one contained in the list.

Table 7 Latency effects on T_2: difference in p_2 features calculated from local timestamps compared with features derived from respective network timestamps (ms)

Subjects	Min	Max	Average
N_1	2	233	67
N_2	2	73	27
N_3	0	54	16
N_4	2	456	44
N_5	1	250	52

5.2.7 T_3 discussion

In comparison with T_2, we can see a lower classification performance for the encrypted data stream. Based on the results shown in Fig. 6, we can infer that an increased length of the search string leads to increased verification performance and a lower EER. Hence, the best performance is achieved for p_2 with an EER value of 23.61%. We can also see that, at least in our experiment, for different search string lengths, the overall performance will be similar to the performance of evaluation of the shortest search string. However, due to our limitation to two different search strings, we cannot quantify this finding.

The analysis of the identification partially confirms the assumption that the experimental setup is influenced by the differences in the utilised hardware. At least two test subjects shared a common computer for recording the sessions—during the identification of the samples from those two subjects, usually both have yielded highest similarities with the stored templates. Nevertheless, mostly the correct decision was made by the identification algorithm.

6 Conclusion

Due to the nature of biometrics, data is generated at every step we take in an online environment. Combined with easily accessible processing power and algorithms, it is easier than ever to identify or track people by methods they may be totally unaware of. The behaviour of each search engine may differ, but we have demonstrated the principle potential of recognising users of SSF, even in scenarios where data is incomplete. We only consider the data that is transmitted independently yet have demonstrated the capability and risks. We have demonstrated the overall approach both under consideration of decryption of single packets (using mitmproxy) and by considering only encrypted packets, the latter using a simple Wireshark network tool. In both cases, we filter client outgoing network streams to search engine providers and extraction of metadata to build the biometric features for identification and/or verification purposes.

Without clear consent, biometric data is given and could be collected for each user and processed for tracking purposes. As further data can be gathered, this would only increase the ability to identify people, either through particularly distinct phrases with high intra-class variation or through methods such as concatenation as shown.

Our first proof-of-concept experiments indicate that biometric recognition is feasible with an EER of up to

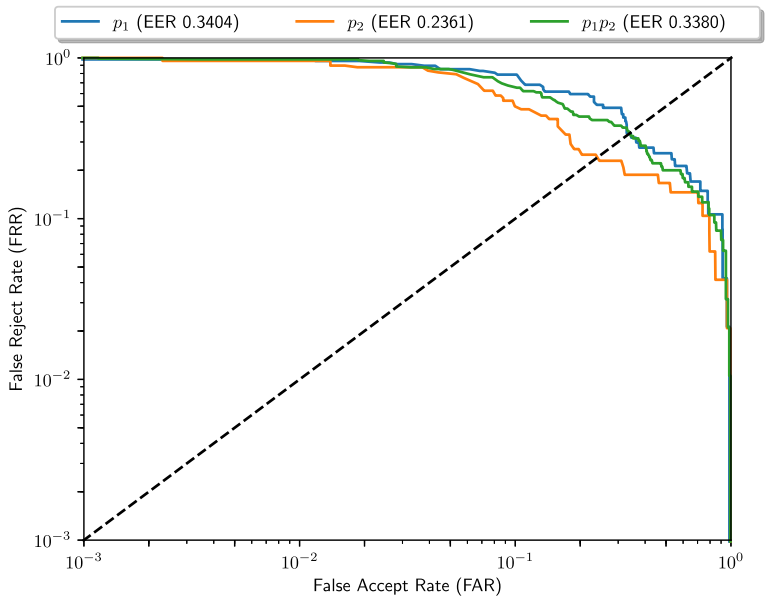

Fig. 6 DET graph for $T_{3.1}$ (lower EER values indicate a better performance)

5.11% in the non-encrypted stream, e.g. by the operator of a search engine, and 23.61% for the encrypted stream, which is visible to any network node transferring the search requests. This is especially worrisome since information like monitor resolution, operating system, and other hardware details can already be captured by web service providers. This information can be exploited along with an IP to track the machine used. In light of keystrokes used as an identifier on top of these established methods, even in a scenario with several people in one household sharing use of the same device, the search provider may learn to distinguish users and, therefore, learn which search terms are searched by each such user. To this end, the new interesting field of biometrics in SSF-enabled web systems has been established.

To protect against this potential for tracking, as a basic protection measure, we would recommend disabling JavaScript. This demonstrably disables any suggestion behaviour within the four tested search engine websites, and would ensure the most vulnerable biometric data would not be transmitted. However, this is not a total solution against browser inputs which do not have JavaScript dependencies such as typing in the address bar or through search engine browser extensions. Alternatively, development of some specific browser add-on could offer protection, for example by withholding user input until pressing enter. Note that either of these approaches would also naturally disable the SSF outright.

We have only considered the basic scenario of a fixed-text keystroke system to examine the possibility of identification through network packets generated from typing behaviour. As we have demonstrated such capability, we recommend further work to inspect the potential beyond fixed-text to free-text analysis. This would explore the risks of whether previously unrecorded search terms could be used to identify people and therefore would cover even broader scenarios of identification from searching behaviour.

There is great scope for future work in this field of biometrics. More extensive investigation using more subjects in similar experiments could uncover potential for identification of larger user groups. Further experimentation with pre-processing, such as utilisation of discarded incomplete samples as substrings of phrases, or adjustment of the approach to missing values and/or interpolation, could improve results. Different classification approaches and advancements in machine learning could be applied beyond the NB and NN classifiers presented, and additional examination and experimentation with control over network conditions is warranted. For the template, matching additional features could be evaluated. Beyond that, further header data could be evaluated in order to minimise artefacts in the extracted sessions. Moreover, the application of the methods presented in

Table 8 Identification performance in the encrypted data stream $T_{3.2}$ (value range [0,1]; higher values indicate better results)

Rank-Level	p_1	p_2	p_1p_2
rank-1	0.2553	0.3541	0.3895
rank-2	0.4255	0.6458	0.4526
rank-3	0.6170	0.7916	0.5684

this article should be investigated further for other services that use user input, e.g. for cloud-based office solutions. Another field of further investigations might be the de-anonymisation of users in the TOR network, specifically when observing the entry node traffic.

Abbreviations

K_{HW}: Hardware keyboard (Linux); K_{SW}: Software keyboard (Android); EER: Equal error rate; QA: Quality assurance of packet contents; SSF: Search suggestion functions; *B*: Browser; *K*: Key input method; *S*: Search engine; *V*: Virtual machine

Acknowledgements

The work presented has been supported in part by the European Commission through the MSCA-ITN-ETN - European Training Networks under Project ID: 675087 ("AMBER - enhAnced Mobile BiomEtRics"). The information in this document is provided as is, and no guarantee or warranty that the information is fit for any particular purpose is given or implied. The user thereof uses the information at one's own sole risk and liability. We thank all the anonymous volunteers for contributing test data to experiments T_2 and T_3.

Authors' contributions

JD originally proposed the idea and MH performed the initial speculative tests. All authors contributed in the discussion of the topic and the overall approach. JD, KL and SEG each gave input during the project and provided their experience to aid NW, NK and KJ in their implementation, testing, and reporting of T_1 and T_2. MH implemented the algorithms and performed the tests for T_3 with all authors performing analysis of the test results. JD and CV both gave significant guidance on the motivation section and the article structure, while all authors contributed in writing and approved of the final manuscript.

Funding

The work presented has been supported in part by the European Commission through the MSCA-ITN-ETN - European Training Networks under Project ID: 675087 ("AMBER - enhAnced Mobile BiomEtRics"). This project has received funding from the European Union's Horizon 2020 research and innovation programme.

Availability of data and materials

The keystroke data sets collected and analysed during the current study are not publicly available due to the sensitivity of biometric data. Volunteers had privacy concerns and did not consent to wider data sharing (as is their right in GDPR).

Competing interests

Two of the guest editors for this submission, Richard Guest and Christian Kraetzer, are members of the European Training Network AMBER, as are authors NW, SEG, JD and CV. Furthermore, Christian Kraetzer is also a member of the AMSL workgroup at OVGU, which is represented by all authors.

Author details

[1]Multimedia and Security Lab (AMSL), Otto-von-Guericke-University, Magdeburg, Germany. [2]Department of Informatics and Media, Brandenburg University of Applied Sciences, Brandenburg an der Havel, Germany.

Received: 23 August 2019 Accepted: 14 January 2020
Published online: 21 February 2020

References

1. European Parliament and Council, Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 (General Data Protection Regulation), Official Journal of the European Union, L 119, 1-88 (2016)
2. K. Haring, *Ham Radio's Technical Culture, 1st edn*. (MIT Press, Massachusetts, 2006), p. 23
3. R. S. Gaines, W. Lisowski, S. J. Press, N. Shapirog, *Authentication by keystroke timing: some preliminary results, Rand Rep. R-2526-NSF*. (RAND Corporation, California, 1980)
4. F. Monrose, A. Rubin, in *Proceedings of the 4th ACM Conference on Computer and Communications Security, 4CCS97, Zurich, Switzerland, April 1-4, 1997*. Authentication via keystroke dynamics, (1997), pp. 48–56. https://doi.org/10.1145/266420.266434
5. P. H. Pisani, A. C. Lorena, A systematic review on keystroke dynamics. J. Braz. Comput. Soc. **19**, 19–573 (2013). https://doi.org/10.1007/s13173-013-0117-7
6. D. Buschek, A. D. Luca, F. Alt, in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI'15, Seoul, South Korea, April 18-23, 2015*. Improving accuracy, applicability and usability of keystroke biometrics on mobile touchscreen devices, (2015). https://doi.org/10.1145/2702123.2702252
7. C. Rathgeb, A. Uhl, A survey on biometric cryptosystems and cancelable biometrics. EURASIP J. Informa. Secur. **2011**(1), 3 (2011). https://doi.org/10.1186/1687-417X-2011-3
8. R. Koch, G. D. Rodosek, in *Proceedings of the 6th International Conference on Network and Service Management, CNSM 2010, Niagara Falls, Canada, October 25-29, 2010*. User identification in encrypted network communications, (2010), pp. 246–249. https://doi.org/10.1109/CNSM.2010.5691292
9. A. Cortesi, M. Hils, T. Kriechbaumer, contributors, mitmproxy: a free and open source interactive HTTPS proxy [Version 4.0] (2019). https://mitmproxy.org/. Accessed 6 December 2019
10. D. Hosseinzadeh, S. Krishnan, Gaussian mixture modeling of keystroke patterns for biometric applications. IEEE Trans. Syst. Man Cybern. C (Appl. Rev.) **38**(6), 816–826 (2008). https://doi.org/10.1109/TSMCC.2008.2001696
11. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)

Publisher's Note