This Is Your Behavioral Keystroke Biometric on Rubbish Data

Roy Maxion **Vrishab Commuri**

September 2020
CMU-CS-20-134

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Sponsors: National Science Foundation, grant number CNS-1319117; and CMU Software Engineering Institute.

Keywords: Keystroke dynamics, behavioral biometrics, data quality.

Abstract

Keystroke dynamics is a behavioral biometric, typically used to identify people based on their typing rhythms, and to distinguish between legitimate and fraudulent users/behaviors. Two common sources of typing data are: controlled laboratory environments; and real-world, field environments. Because keystroke researchers tend to use lab and field data interchangeably, it is conjectured that any differences between lab and field data are nil or trivial, the effects of which can be ignored at no cost to an automated decision-making procedure that distinguishes between legitimacy and fraudulence. We test this conjecture by conducting a lab-based typing experiment, and replicating it under field conditions, each with 100 participants. The lab environment used a single hardware/software platform and keyboard with high-resolution keystroke timing, whereas the field environment relied on whatever hardware and keyboard a volunteer participant happened to have. An analysis of both sets of typing data revealed that USB keyboards, used in the field, injected artifacts into the data, causing the data to lack fidelity to the actual keystroke signal. These artifacts were observed to change an algorithm's decision by nearly 20 percentage points, wrongly reversing a distinction between legitimacy and fraudulence. This paper chronicles the methods by which these artifacts and their damaging effects were discovered.

1 Introduction

Keystroke biometrics (or keystroke dynamics) is the term given to the procedure of measuring and assessing a user's typing style, the characteristics of which are thought to be unique to a person's physiology, behavior and habits. Ascertaining the unique typing style attributed to a given user through their typing rhythms is an idea whose origin lies in the observation (published in 1897) that telegraph operators have distinctive patterns of keying messages over telegraph lines [4]. These patterns came to be known as the "fist" of the sender, due to the interaction between the operator's fist and the knob on the telegraph key.

One crucial aspect of fists is that they emerge naturally, as noted over a hundred years ago by Bryan and Harter [4], who showed that operators are distinctive due to the automatic and unconscious way their personalities express themselves, such that they could be identified on the basis of having telegraphed only a few words. Hence both keying styles and typing styles are natural behavioral biometrics, so called because they're based on behavior unique to an individual.

Keystroke dynamics relies on the timing of key-presses and key-releases as text is typed on a keyboard. The fundamental measurements are the moments in time of key-down and key-up behaviors. These measurements are the basis for deriving features for analysis: hold time (how long a key is held down), down-down (dd) latency (from one key-down event to the next key-down event), up-down (ud) latency (from one key-up to the next key-down), and other features that arise similarly from combinations of the basic key-down and key-up signals.

These measured features are compared to a user profile as part of a classification procedure. A match or a non-match can be used to decide whether or not the user is authenticated, the user is legitimate or fraudulent, or the user is the true author of a typed sequence. Keystroke dynamics, as a technology for facilitating this match/non-match process, has recently been endorsed for strong customer authentication by the European Banking Authority, covering the European Union [11].

For research purposes, keystroke data can be gathered in laboratory or field (real-world) conditions. Lab data can be more difficult to obtain, because participants need to be recruited and scheduled for lab time; field data, on the other hand, can be gathered over the Internet, either in custom web deployments or through crowd-sourcing systems such as Amazon's Mechanical Turk. When the same experimental protocol is used in both lab and field situations, there should be no difference in the data collected. If it's true that there is no difference, then it makes sense to conduct such studies in the easiest and simplest way possible. So, which data set, if either, would be better – lab or field, and why? The answer is surprising and distinctly unintuitive.

2 Problem and approach

The problem addressed in this paper is to characterize the differences, if any, between a lab-based data set and a field-based data set, each collected under the same experimental regime, other than the data-collection venue – lab vs field.

- Research Question 1: What distinguishes lab vs field data?
- Research Question 2: If distinctions exist, do they matter?

Our approach began with what many fields would regard as a very standard methodology: exploratory data analysis (EDA), comprising the ordinary application of descriptive statistics, combined with graphical analysis in a way that, as Tukey said, "forces us to notice what we never expected to see" [32, pp. vi].

We first explain how we acquired our lab and field data. We engage exploratory data analysis and descriptive statistics to get a sense for the data, which reveals an unexpected discrepancy that bears a deeper look. Then we examine the data graphically, and see that there are striking differences between lab and field data for which there is no obvious explanation. We examine carefully the critical differences between lab and field operations. This leads to scrutinizing the mechanisms intrinsic to keyboards, and a detailed statistical and graphical confirmation that some keyboards play a critical role in data integrity. Finally, we show that artifacts injected into the data by certain keyboards make a difference when using the data for making decisions about fraudulent vs legitimate user behavior. Finally, we discuss the results, followed by a note about the limitations of this study, and the consequences of the discoveries made here.

This paper chronicles our journey of exploration, culminating in a surprising discovery, which may change how people think about data-gathering instrumentation in keystroke dynamics.

3 Related work

Two kinds of related work are relevant here: keystroke dynamics in general, and keystroke dynamics data quality/integrity. We review these in turn.

General keystroke dynamics. We have already described the basic tenets of keystroke biometrics/dynamics in Section 1, and won't repeat those details here. For readers seeking a general currency in keystroke dynamics, a number of surveys and reviews are available. Although the ones cited here appear dated, they are still quite relevant; perhaps there are no very recent surveys because the ones noted here (in chronological order) have served the community well.

• Peacock et al. [28] provide a general and very accessible overview of keystroke dynamics; for those unfamiliar with the technology, it's a good place to start. The authors presented results for 23 different keystroke systems; however, some of the more impressive systems may have achieved their results due to small sample sizes and methodological aberrations.

• Banerjee and Woodard [1] wrote one of the more thorough surveys available in the literature, citing 162 references. The survey covers a range of pertinent issues such as the typing environment, biometric features, metrics, five publicly-available databases, statistical and pattern-recognition algorithms, performance-shaping factors, and commercial applications. The paper also addresses related issues such as the underlying psychology of typing, data-acquisition environments, and several nice tables comparing a range of keystroke studies on the basis of the algorithms used: statistical, neural networks, pattern recognition and learning, and heuristics. The paper compares systems from 77 studies; however, almost none of these systems were evaluated on the same data set, so some of the results could have been due to the particular constituency of the typists.

• Teh at al. [31] provide an insightful survey of keystroke dynamics covering nearly three decades of research. They note the basic aspects of the field, experimental protocols, data acquisition, features, feature selection, classification approaches (which are covered particularly well),

and comparisons amongst keystroke studies based on various inputs.

• Zhong and Deng [33] provide a broad-ranging survey of keystroke dynamics aspects including features, classification techniques, distance metrics, keystrokes on mobile devices, emotion detection, and eleven benchmark data bases. They commented on a lack of a common evaluation framework for keystroke dynamics (which continues to be true, even now).

• Obaidat et al. [25] wrote a terse, but competent survey that covers most of the important aspects of keystroke dynamics: basic mechanisms and metrics, comparisons with other biometrics, evaluation measures, applications, features (for analysis), feature selection, classification methods, and a compendium of benchmarking data sets.

The progression of research on keystroke biometrics/dynamics is sadly lacking in scientific advancement. While there has been progress in terms of applying new (and relevant) classification, anomaly-detection and clustering algorithms, these have largely been applied to single (and often inadequate) data sets without a sound basis for many of the claims made. The discipline's story certainly leads to a greater variety of approaches and techniques, but seldom to a better fundamental understanding. The same can be said about the quality or integrity aspects of the field.

Data quality. The following papers treated, chronologically, various aspects of data quality in keystroke biometrics. This issue is of particular concern in the present paper because data quality, or rather data integrity – the faithfulness of the data to the original signal – lies at the heart of our investigation and our results. In the extant literature there is no mention of such a concept; most of the cited papers here refer to data quality, but never define the term. Because the earliest keystroke papers were written in the 1970s, it's surprising that scant attention has been paid to a topic presumably so important. Nonetheless, we note some of the papers that at least mention quality.

• Kang et al. [15] addressed data quality in terms of uniqueness and consistency of typing. They opined that data are of higher quality when a user's typing is more unique (more distinct from others' typing), and when a user's typing is self-consistent (less variability within a user). They introduced a scheme of artificial pauses and cues (e.g., metronome) that typists could follow – like providing different rhythms to different users, thereby ensuring a greater variability between users and a lesser variability (uniformity of typing) within users. It is easy to see how this could improve classification performance; however, their results did not report classification accuracies, so the practical outcomes of such schemes are unknown.

• Rybnik et al. [29], in a paper entitled "The Practical Impact of Database Quality," discuss the quality aspects of two keystroke data sets that they examined across several dimensions, including consistency and accuracy; one data set was a laboratory benchmark (from CMU), and the other data set was collected over the Internet, using Javascript and a web browser. No specific definitions of quality were offered, although aspects such as accurate timing were hinted at.

• Giot et al. [12] evaluated and ranked a set of benchmark data sets for keystroke dynamics. Although their paper wasn't intended as a missive on data quality, it gives substantial attention to the concept, including a number of criteria that affect data quality, such as the type (shape) of the keyboard used, the operating system, clock resolution, acquisition environment (e.g., lab vs field) and the data-gathering protocol itself. Electrical or communication characteristics were not mentioned.

• Sun et al. [30] concentrate on shared data sets and their quality, although quality is not explicitly defined, nor are there clear criteria for assessing data quality. They note that "there are only a few benchmark data sets of high quality," citing [12, pp 2], whose data are examined later in this paper. They observe a number of data-collection practices that are certainly worthy of attention, but without much explanation. They develop and evaluate a new data set of their own, but without specifics, so it's hard to learn what's important in terms of quality.

• Pavaday et al. [27] noted that different operating systems can have different effects on the timing accuracy of keystrokes. They did a detailed investigation of timing mechanisms, concluding that not all timers produce the same (or even repeatable) results. However, they did not look at the effect of the keyboard itself, nor its method of communication with the operating system.

In closing we note that there is really no work that we know of that bears on data integrity or quality, other than to give these terms a fleeting mention. Given the longevity of the field, this is a pity, as surely the matter of quality or integrity would be of some interest. In the following sections we show how detrimental it can be to ignore these matters.

4 Methods

This section provides the details of how our investigation was conducted, so that readers can judge (a) whether or not our methods were appropriate, and (b) whether or not our procedures and results are reproducible.

4.1 Data acquisition

4.1.1 Task

Subjects typed a 10-character password-like string 50 times in each of 8 sessions. The 8 sessions were separated by at least one day, for a total of 400 typed repetitions of the password over 8 days.

4.1.2 Materials

The password-like string typed by the subjects was `.tie5Roanl` followed by a carriage return, which was also recorded. This string was meant to resemble a strong password (documented in [17, section 4.1]).

4.1.3 Subjects

Our goal was to have 100 lab and 100 field subjects. Subject recruitment was over-subscribed in anticipation of drop-outs due to not finishing the task, not following instructions, data corruption, etc. After data pruning, 200 subjects participated: 100 in the lab condition (45 female, 55 male) and 100 in the field condition (46 female, 54 male); all signed an IRB-approved consent form. Lab subjects were recruited from university environs by poster, recruiting service, and word of mouth; field subjects were recruited similarly, but from around the world, anywhere on the Internet. No compensation was provided. Subjects were required to be at least 18 years old, be fluent in English, have at least three years of experience typing on a computer, and type at least 30 words per minute.

4.1.4 Instructions to subjects

Subjects were instructed to type normally. They were informed that the task was neither a speed nor an accuracy test (this was necessary because in pilot studies we discovered that many subjects believed they were being tested for speed or for accuracy or for both, under which conditions their typing would not be normal).

4.1.5 Environment

Our experiments were carried out, in accordance with the exact same protocol, in two environments: lab and field. The laboratory environment was tightly controlled in nearly every regard: lighting, temperature, computer, keyboard, timers, furniture (e.g., ergonomically adjustable chair and table), ambient noise, etc. The field environment was "out there" anywhere on the Internet, using whatever Windows machine (and operating system version) was available to a subject, along with that person's preferred keyboard, monitor, ergonomic conditions, etc.

4.1.6 Data features and measurements

The measurements taken were only keystroke timing: the moment in time when a key was pressed, and a somewhat later moment in time when the same key was released. From these press and release times we can derive the primary keystroke-dynamics features: (a) hold time – the amount of time a key was held down, typically ~90-100 milliseconds; (b) down-down (dd) latency time – the amount of time between a key-down event and the immediately following key-down event, typically ~220-230 milliseconds; (c) up-down (ud) latency time – the amount of time between a key-up event and the immediately following key-down event, typically ~130 milliseconds.

4.1.7 Apparatus & instrumentation

The equipment differed slightly between lab and field conditions. In the lab we used an IBM ThinkPad X60s notebook computer (type-model 1702-4EU, 1.5 Gb RAM, operating system Windows XP Professional, Service Pack 2); a Dell UltraSharp 1907FP 19-inch external monitor, and an Apple M9034LL/A external keyboard. The Apple keyboard was modified internally; its keyboard encoder was completely bypassed, and the key matrix was scanned by a custom encoder which tagged key events with values from an on-board timer whose worst-case, calibrated resolution was 100 microseconds. Networking was turned off to minimize load on the XP machine. Details of the equipment, the timer and how the timer was calibrated are in [21]. The equipment used in the field condition was whatever Windows machine (OS version 7 or above) and keyboard the volunteer subject happened to have.[1] Details of field apparatus (e.g., OS, version number, hardware, CPU, etc.) were logged automatically.

A custom user-interface presented the string to be typed (`.tie5Roanl`) in a small dialog box; the typing was entered into an accompanying text input box. Typing errors were detected and

[1] Restricting the experiment to Windows machines was not an impediment, because – according to Statistica – Windows had 77.61% of the operating system market as of November 2019; see: `https://www.statista.com/statistics/218089/global-market-share-of-windows-7/`.

discarded; subjects were asked to repeat a mistyped entry so that every subject would complete 50 error-free repetitions of the "password". This same user interface collected the keystroke timing data via the Microsoft QPC timer. The user interface was exactly the same in both lab and field conditions, except that the keystroke timing data for the lab condition was collected from an external, custom timer, as noted above.

4.1.8 Procedure

The procedure for lab subjects and field subjects was exactly the same. Subjects started by signing an IRB-approved consent form, filling out a demographic questionnaire, and reading instructions. The typing task started, the subjects typed 50 repetitions of the "password," read a debriefing statement, and departed. Lab subjects returned for the remaining 7 (of the 8) sessions, usually skipping a day between sessions. Field subjects were instructed to do the same, although some people had intervals longer or shorter than every two days. No one participated in more than one session in the course of one day, and no subject in one data set was in the other data set; no overlap.

4.2 Exploratory data analysis (EDA) / descriptive statistics

Here we begin our description of how the typing data were analyzed to discover any differences (or lack thereof) between lab and field data. A typical way to begin exploring a data set is with a statistical summary, which provides a concise view of the location, spread and range of the data.

Table 1 shows descriptive statistics for hold times in lab and field data sets. Note that many of the numbers are in rough agreement from one data set to the other, except for kurtosis; here the field data depart significantly from the lab data, an observation that suggests a big difference in outlier behavior, begging further investigation.

	Lab	Field
Mean	92.289	99.985
Std	30.436	42.842
Median	88.400	95.900
Mode	80.500	80.000
Kurtosis	42.388	7997.751
Skewness	1.693	51.278
Range	2033.900	8687.900
Max	2035.300	8688.200
Min	1.400	0.300
Inter-Quartile Range	38.600	37.900
Upper Quartile	109.300	116.900
Lower Quartile	70.700	79.000

Table 1: Descriptive statistics: hold times, in milliseconds: 11 features (keys), 50 repetitions per key, 8 sessions per subject, 100 subjects; 440,000 data points for each data set, lab and field. The mean is the average time that a key was held down; the difference between the lab and field means is significant ($p < .001$, two-tailed t-test). Note the huge lab/field difference in kurtosis.

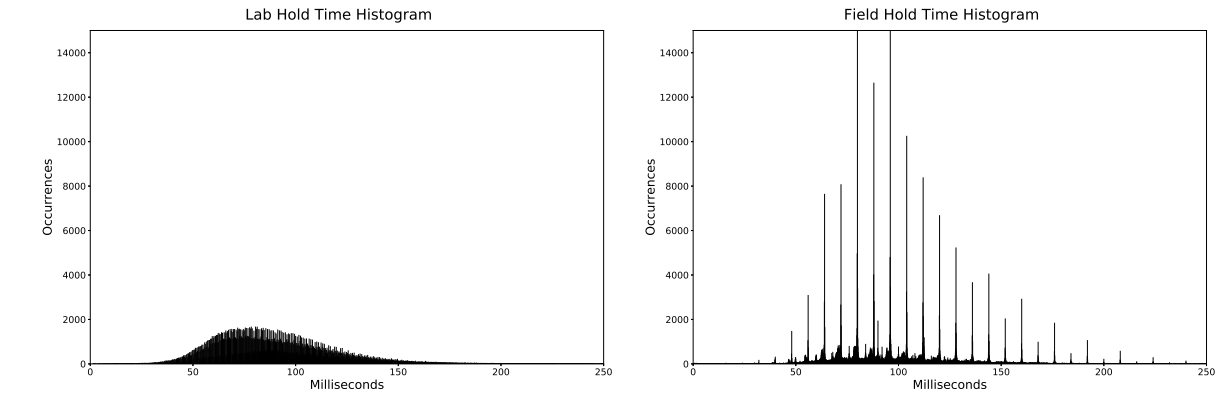


Figure 1: Lab hold times (left); field hold times (right). Field-data spikes are 8 ms apart.

4.3 Graphical exploration; plotting the data

One important tenet of exploratory data analysis [32] is "plot the data." We plotted the number of instances in which a keystroke hold-time occurs at a given time interval in milliseconds – e.g., how many times did an 80-msec hold time occur? Figure 1 shows the results. As we can see in the figure, the field data contain a number of tall spikes. These spikes, which demand explanation, are 8 milliseconds apart. There are no such spikes in the lab data. Why?

4.4 Comparative frequency counts of selected events from EDA

Figure 1 shows that some field-data hold times are much more prevalent than others, and not apparent at all in the lab data; see the tall spikes in the right panel. Although neither data set should exhibit these spikes, we'd at least expect them to be roughly equally distributed across lab and field data. Table 2 helps explore these discrepancies between lab/field hold-time prevalences, where we see that the top 10 most frequently occurring counts of field-data hold times (in ms) are far higher than those for lab data, but again one wonders why. To explore further we turn again to exploratory data analysis – scattergrams.

Lab		Field	
Holds	Counts	Holds	Counts
80.5	1656	80.0	15250
81.3	1654	96.0	15094
78.4	1653	88.0	12630
77.6	1627	104.0	10239
75.5	1613	112.0	8371
71.8	1608	72.0	8061
84.2	1600	64.0	7628
76.0	1592	120.0	6669
76.8	1587	128.0	5216
85.5	1585	79.9	4949

Table 2: Top 10 hold-time frequency counts (ms) for 100 lab and field subjects. Field-data frequencies are larger; they occur at 8-millisecond intervals (except 79.9), while lab data do not.

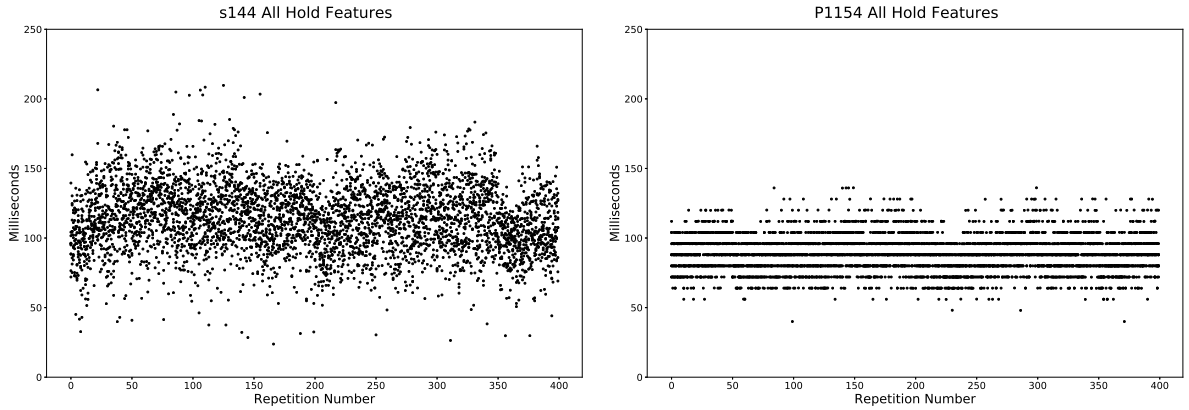


Figure 2: Comparison of lab hold-time data (subject 144, left) and field hold-time data (subject 1154, right). The lab scattergram on the left is to be expected, with points scattered roughly evenly throughout the plot. The field scattergram on the right is comprised of horizontal striations, vertically separated by 8 ms. Each plot shows 8 sessions of 50 repetitions each, for 400 typed repetitions total; session boundaries are indicated by the ticks on the x-axis.

4.5 Graphical analysis

Figure 2 shows plots for two subjects – one typical lab and one typical field – where the lab subject's hold-time scattergram is what one might expect to see, but the field scattergram has surprising, and unexplained, horizontal striations across it[2]. The striations appear to be 8 milliseconds apart on the y-axis (later confirmed). The field subject depicted in the figure was typical of about 90% of field subjects, although not all of them had striations separated by 8 milliseconds; others appeared to have striation separations of 2, 4, 16, etc. milliseconds, as well as other separations that were not definitively on power-of-two boundaries. Some of the field data seemingly contained no striations at all, and some subjects showed striations in some, but not all, of their 8 sessions. The lab data, however, were completely devoid of striations.

4.6 Explaining the graphical striations in the data

There were no striations in the lab-data plots, but most of the field data did contain striations (cf. Figure 2). Since both lab and field protocols were exactly the same, including the same application software, we look to what was otherwise different between the two environments. The differences lay in the hardware and software platforms, the timing regimes in the two environments, and the keyboards used. We examine each of these in turn.

• Hardware platform. Our field app logged a user's machine type, operating system and version, and processor type and speed. Machine types had a wide range, including Dell, HP, Lenovo, Asus,

[2]We use hold times because they are said to be more discriminative than latency times: [9, 26]. Latency plots show similar behavior, but would be redundant here.

Mac[3], Sony, etc. The processors also varied (Intel, Pentium, AMD) and reflected a broad range of CPU speeds and clock frequencies. Because the phenomenon of striations persisted across all of these hardware platforms, it seems reasonable to conclude that the hardware was not causative.

• Software platform. Two elements comprised the software platform: the keystroke acquisition software, which was exactly the same for both lab and field; and the subjects' operating systems. The field operating systems spanned several versions of Windows, with a range of service packs. The operating system in the lab was also Windows. Because the same kinds of striated plots were observed irrespective of operating system/service pack, it is concluded that the particulars of the operating system couldn't be responsible for the striated plots. Some subjects, e.g., field subject 1106 in Figure 3-a (cf. page 12), used the same operating system for all experiment sessions, but only some of those sessions were striated, or quantized, while others were not. Some subjects showed no quantization at all. The software platform appears not to have been a factor in whether quantizations appeared or not.

• Timing regime. As previously noted, the lab system used a custom, out-of-band timing device with 100 microsecond resolution. The field timing system relied on the Microsoft Query Performance Counter. The QueryPerformanceCounter (QPC) value, when scaled with the value from QueryPerformanceFrequency, provides high-resolution time-interval measurements, independent of the speed of the CPU clock [22]. It is this QPC mechanism on which we base the duration of hold and latency times. While the difference in resolution between the lab and field timing regimes can influence classification accuracy [16], there's no reason to think that either of these timing mechanisms would induce striations in the data. Consequently, even though different hardware platforms used different CPUs running at different speeds, the effect of CPU differences on QPC-based time stamps would be negligible, and certainly would not induce the striated behavior seen in the plots in Figure 2.

• Keyboard. Subjects in the lab used only a single keyboard (cf. Section 4.1.7), so there was no keyboard variability at all in the lab. In the field, subjects used whatever keyboards they had at hand, as noted in our demographic logs: standard US keyboard, natural or Kinesis ergonomic, Mac, etc. Unfortunately, we had no definitive way to determine keyboard make and model, although some users reported make/model out of band. We examined the critical aspects of keyboards (scan matrix, switches, keyboard encoder (sometimes termed microcontroller), debouncing, etc.), but could find no differences that would explain the quantizations in the field data. While there are minor distinctions amongst keyboards, none of them are likely to make any difference to the keystroke monitoring software; these distinctions are ergonomic preferences, with some people favoring a particular keyboard layout, key type, key travel, haptic feedback, function keys, etc.

While the physical keyboard itself may matter little, what might matter more is the way that the keyboard communicates with the host. There are two predominant communication protocols, USB (Universal Serial Bus) and PS/2 (Personal System/2), that specify both the electrical signals and data format from keyboard to host. For the lab data we used an Apple keyboard (cf. Section 4.1.7) whose internal electronics were removed and replaced with a custom circuit for scanning the key matrix and capturing timing at a resolution of 100-microseconds. In the field, people used whatever

[3]Mac users ran the Windows-based typing software via Bootcamp, a Mac utility that allows switching between MacOS and Windows.

keyboard they happened to have, most of which (but not all) were USB-based.

Exploring the above-noted differences in lab/field apparatus, it is instructive to look ahead, to our exposition on "unruly" data – data that didn't conform to expectations – as shown in Figure 3-a (page 12). Field subject 1106 used a Lenovo laptop (model Y510p, Windows 7, Professional Service Pack 1 (Build 7601)) for every one of his eight daily typing sessions. Striations are evident in sessions 1-4 and 6, but sessions 5, 7 and 8 resemble the lab data shown in Figure 2, left panel. Whatever might be causing the striations, it apparently wasn't the computer, the operating system, or the monitoring software, because we observe behavioral changes in the data without changing any of those factors. However, this subject did not use the same keyboard for all sessions; he used the laptop's internal keyboard when he was traveling (sessions 5, 7, 8), and a wired Logitech "Wave" keyboard when in his office. Other than ergonomics and convenience, the only difference between these two keyboards was the way they communicated with the host computer: USB (external) or PS/2 (internal). This observation led to a comparison of the USB and PS/2 protocols.

The USB specification explains how the USB stack works [5]. At a very high level, when a USB device is plugged into a computer, the operating system initiates a conversation with the device to determine what kind of device it is, how fast the device is, what its data rate is, and how often the operating system should request data from the device. This last aspect, requesting data from the device, is called polling. A keyboard is a slow human interface device (HID in the USB specification). Since a keyboard is a slow device (compared to a disk or a streaming device), and not much data are transferred at any one time (i.e., a few keystrokes), it doesn't need to be polled very often; once every eight milliseconds is typical. Coincidentally, an 8-millisecond polling rate corresponds to the 8-ms spikes we observed in Figure 1, suggesting that USB keyboards could somehow be associated with the striations.

PS/2 is a fast, hardware-based, interrupt-driven, serial communication protocol designed by IBM [14]. Due to the keyboard initiating communication within the PS/2 protocol, rather than having the host machine periodically poll it, every key-press (make) and key-release (break) triggers an immediate interrupt to the CPU.

Based on the operational details of PS/2 and USB, it seems reasonable to conclude that the USB stack is responsible for the quantization that we've seen in the field data. To firm up that conjecture, we should verify the polling rates for each of the field subjects, confirming that subjects who used USB keyboards showed corresponding quantization patterns in their data. To do that we need a tool to assess the polling rate for each subject.

5 Polling discovery

Based on the scattergrams we saw, it seemed sensible to think that much of the field data had been influenced by USB polling. To test this notion, we need to know two things: (1) do all subjects' data exhibit striations; and (2) what is the interval (milliseconds) between the striations? That interval should correspond to a USB polling rate. Since we couldn't examine all 800 field sessions manually, we built an ensemble polling discovery tool to extract the information automatically. The results of our tool are presented in Table 3.

Across all 100 field subjects, we constructed a tally of how many subjects had what polling rate. Table 3 shows that the most prevalent polling rate (54%) was 8 milliseconds, followed by 4

Polling Interval	Count	Polling Interval	Count
PS/2	11	5.2	2
2.0	1	5.3	2
2.5	1	5.35	2
2.7	1	5.5	1
2.75	1	7.8	1
2.8	2	8.0	54
3.5	1	10.0	3
4.0	7	16.0	7
5.0	2	22.0	1

Table 3: Polling intervals (msec) for 100 field subjects. 54% polled at 8 msec.

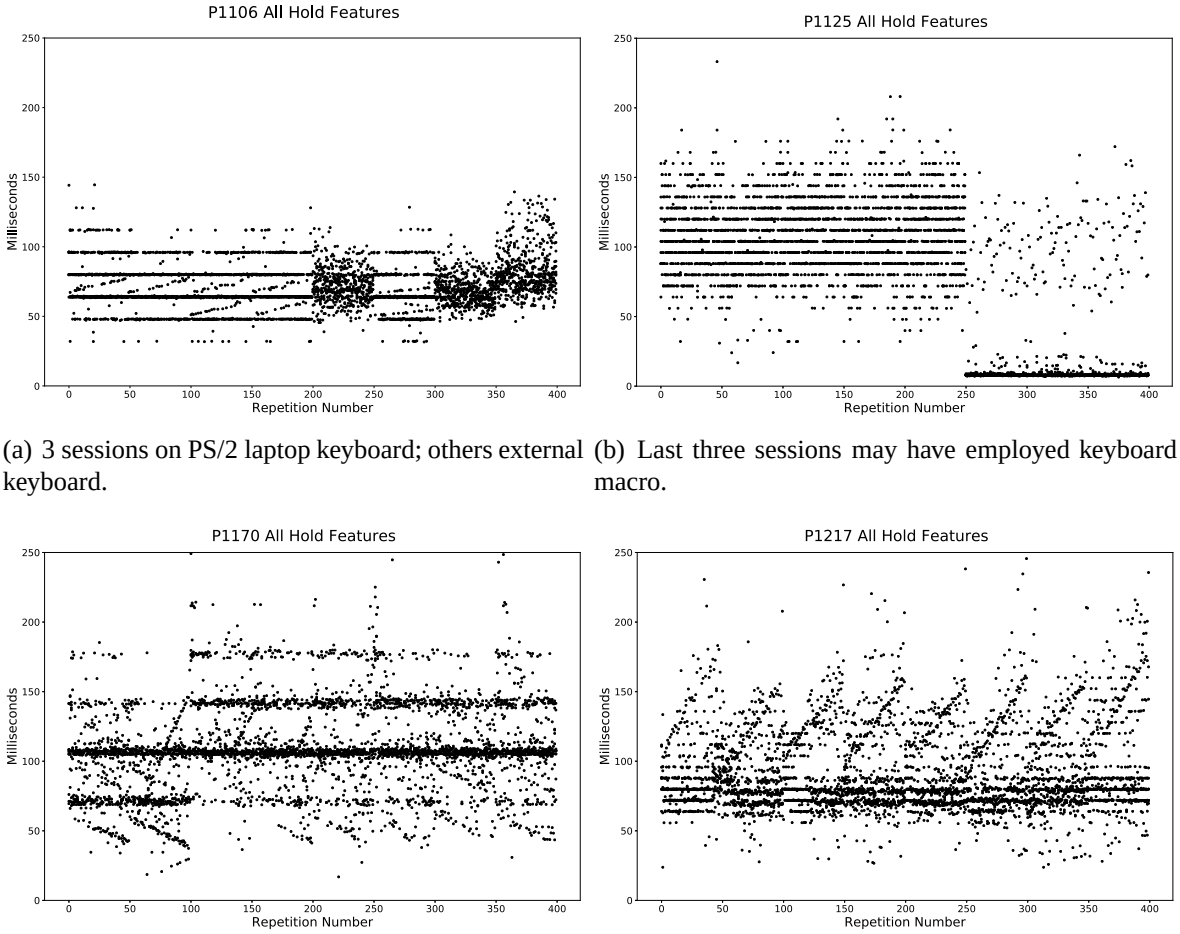
milliseconds and 16 milliseconds, all powers of two; these would be expected, based on the USB protocol. These three polling rates accounted for 68% of the subjects; 11% of the subjects used PS/2 keyboards, based on direct confirmations from 7 subjects, and 4 augmented by discovery-tool results; 6% had an integer polling interval that was not a power of two; the remaining 15% had fractional (e.g., 2.75) polling rates, the cause of which was not ascertained. We speculate that the fractional polling rates arose either from shortcomings in a machine's USB implementation or from particularly inexpensive (hence poorly designed and constructed) keyboards, or both. In applying the polling discovery ensemble to the lab data, no USB polling indications were found.

5.1 Graphical analysis – unruly data

So far, we've seen quantized field data that's been relatively consistent over the course of the eight sessions in which subjects typed; there were no obvious changes or shifts in behavior, as depicted by the scattergrams. However, not all subjects exhibited such regular behavior. Figure 3 shows the scattergrams for four such unruly subjects.

• Figure 3-a shows sessions 1-4 and 6 with the usual striations from field subject 1106, but also shows sessions 5, 7 and 8 with no striations. This subject used the same laptop when working from home during sessions 1-4 and 6; and when working remotely for sessions 5, 7 and 8. During the home sessions he used a Logitech "Wave" external USB wired keyboard; for the remote sessions during travel he used a Lenovo laptop whose internal keyboard was PS/2. The operating system was Windows 7 throughout the course of the 8 days, with no changes. This strongly suggests that the USB keyboard is what caused the striations in the data. In fact, his Logitech keyboard requested 16 msec polling, which corresponds perfectly with his plotted data in the figure; his laptop's internal keyboard was interrupt-driven PS/2 (no polling); hence the disappearance of the striations for sessions 5, 7 and 8.[4]

[4]Some of these details were ascertained through out-of-band communication with the subject.



(a) 3 sessions on PS/2 laptop keyboard; others external keyboard.

(b) Last three sessions may have employed keyboard macro.

(c) 37 ms polling; odd structures may be from insufficient resources.

(d) Old/slow/memory-deprived HP laptop; ramps are 'return' holds.

Figure 3: Four examples of unruly field data; eight sessions, 50 typing repetitions per session. Session boundaries are indicated by ticks on the x-axis. Note the different kinds of irregularities in each panel, as compared to the normal behavior shown in the left panel of Figure 2.

• Figure 3-b shows field subject 1125 polling at 8 milliseconds for his first five typing sessions, and then something changed. It appears that his polling rate dropped to near zero, with the exception of some scattered points in sessions 6-8. The scattered points in the last three sessions depict the subject's `return` key. We don't know what happened in the last three sessions, although we speculate that the subject crafted a keyboard macro that automated his typing, except for the `return` key. Note that the machine in this example was a Mac running Bootcamp.

• Figure 3-c shows field subject 1170, who used a Dell Studio 1749 laptop machine and its internal keyboard; the operating system was Windows 7 Home Premium, 64-bit, with 4 GB of RAM and a 2.13 GHz processor. The polling interval was 37 milliseconds, which we cannot explain. The strange, right-downward-slanting "ramps" in the data are unexplained as well. This

subject reported using streaming music and (occasionally) GoToMeeting, a streaming video and audio application, coincident with typing. Because of these things, we speculate that the machine was starved for resources, perhaps engaged in excessive swapping, which might explain the plot's bizarre appearance.

• Figure 3-d shows field subject 1217, who also exhibits rather odd (and unexplained) behavior, particularly the upward-slanting "ramp" that occurred in each of the 8 typing sessions. This subject ran Windows 10 Home Edition, with a 1.44 GHz CPU, 4 GB of RAM, and an average of 575 MB of free memory at the beginning of each session (the subject reported that the machine got increasingly slow throughout each session), suggesting that this system, too, may have been starved for resources – a probable explanation for the odd-looking plot.

Figure 3's four "unruly" plots illustrate a lesson learned: that not everything goes as expected when conducting keystroke experiments in the field (or maybe even in the lab), and therefore, one should be careful about using all collected data without screening the data first. Plot 3-a showed that different keyboards can produce different data; plot 3-b showed that people may not engage the typing exercise as instructed; plot 3-c showed possible interference from competing processes; and plot 3-d showed that resource starvation may affect the data in ways that don't reflect normal typing behavior. None of the unruly subjects were included in the 100-subject data set.

• Figure 4 shows the typing scattergrams for the same person using two different keyboards in the field experiment (dropped from the study, but used here for illustration purposes). Subject 1172 and subject 1104 are the same person. When the subject identified as 1104, he was typing on a Dell laptop's internal keyboard; the polling rate was 8 milliseconds. When the subject identified as 1172, he was typing on a Windows workstation using a mechanical gaming keyboard (Das Professional Mechanical Keyboard). The gaming keyboard doesn't avoid USB polling, but it polls at one-millisecond intervals. The small interval between polls, combined with jitter[5] from the operating system, makes striations impossible to observe in the scattergram for 1172. Note how different the two plots are, and how much difference the data could make in any classification procedure, as will be seen in Section 8.

In concluding this section we observe that many unanticipated things have the potential to ruin or dramatically affect a data set. There are many varieties of unusual circumstances, including system resources that are available for the typing application, or consumed by other processes, and that users will do unexpected things (e.g., switch keyboards, which sometimes matters and sometimes may not; or employ privacy protection (see footnote 6), which can change the timing of keystrokes), often innocently, and not inform the investigator, because they regard one keyboard to be functionally equivalent to another (and from a user perspective, they're right). The bottom line is: always, inspect/validate the data to ensure that it conforms to expectations.

[5]Jitter can arise from multiple sources; a USB keyboard may not send data *exactly* on bInterval (a USB polling parameter) periods, or USB hosts may not send polling requests *exactly* on bInterval periods (causing early or late key events); or polling rates may be altered slightly, depending on bus traffic.

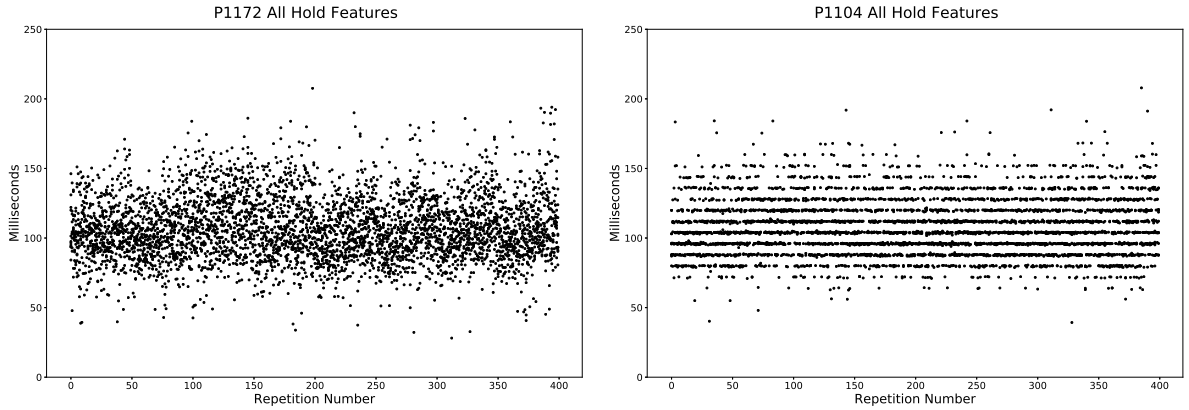


Figure 4: Field experiment, same subject, different keyboards: gaming (left) and USB (right).

6 Other people's data

It's possible that the striations seen in our field data are peculiar to our experiment, our typing application, and our instrumentation. To check this, we looked for similar artifacts in a range of data sets from other researchers. We found USB-induced quantization in all of them; it wasn't just us. Due to space limitations, only two examples are shown in Figure 5.

Giot et al. [12], in a paper that reviewed public benchmark databases for static keystroke dynamics, noted 16 public data sets, some of which were from their own laboratory, and all of which can be freely downloaded. We examined all of the data sets they discussed, except for the CMU data set (which we knew was comprised of data similar to the lab data in the present paper). All of the data that we examined displayed the same characteristics of USB polling that we have seen in earlier figures in this paper. A separate public data set known as BeiHang [18, 19], noted in an international keystroke competition [24], showed similar characteristics. Figure 5 shows subjects from two of the reviewed data sets: BeiHang_B and Web GREYC. We also examined data from a much earlier paper by Bergadano et al. [2], and found the same phenomenon of striations across 44 users. The USB quantization/striation phenomenon stretches far back in time, suggesting that many data sets over the years contain USB-induced artifacts.

The BeiHang_B data set (Figure 5 left) contains 68 subjects in their Data Base 1, with the data for each subject split into training and testing sets. Each subject typed a password unique to the subject five times for the training set, and a variable number of times for the testing set. In order to maintain a consistent number of typing repetitions from each subject, we present results derived from only the training data. Subjects used their own computer and keyboard to type into an online form. The data were collected online. The polling intervals for data set B vary from subject to subject, which might be expected of data recorded from many different computers and keyboards. The data show the striations that are now recognized as being induced by the USB stack.

There are 5 repetitions per subject, with 68 total subjects, so the plot shows 68 * 5 = 340 total repetitions. All subjects are concatenated across the horizontal axis of the plot. Notice that there

are some clearly defined breaks in typing pattern, but not as many as the 68 breaks one might expect. This is probably because some subjects who contributed to the data set shared computer and keyboard with the subject that preceded them (e.g., repetitions 255-275 have similar striations).

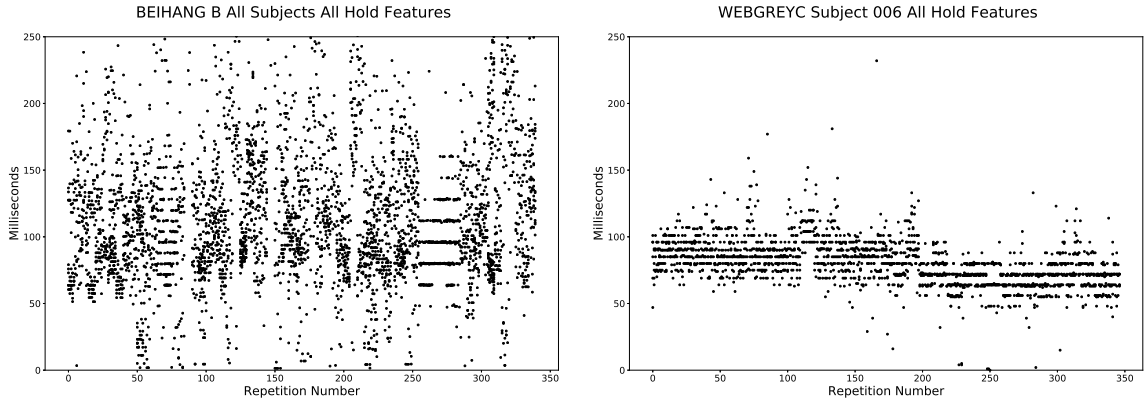


Figure 5: Other people's data – left, 68 subjects, many different keyboards and quantizations, from the Beihang_B data set [18]; right, user 006, 35 sessions from the Web GREYC project – change in quantization at about repetition 100, due to change of computer (Windows to Linux, based on log data); possible keyboard change at about repetition 200 [13].

The Web GREYC data (Figure 5 right) were gathered via a web-based application written in JavaScript, HTML, and CSS. Users were allowed to use any browser, though the majority (54.4%) used some version of Firefox for all sessions; 75.4% used Firefox in at least one session.[6] Subject 006, depicted in the figure, was chosen arbitrarily as a representative subject from the data set. Subjects were asked to complete one session per week, with each session comprised of 10 repetitions of a login and password that are the same for all subjects, 10 repetitions of a login and password chosen by each subject, and 5 repetitions each of the login and password that were chosen by two other subjects (10 repetitions total). Note the changes in polling intervals at approximately repetition 100; this is caused by the subject having changed from a Windows to a Linux machine (while still using the same version of Firefox) mid-way through the data collection. This information was obtained from the Web GREYC log files.

In addition to the aforementioned data sets, we also examined a data set from the Coursera authentication sequence, provided by Dehaye [6, 7], as well as data from Dhakal et al's [8] paper describing observations on typing 136 million keystrokes. Although we don't present the corresponding scattergrams, they contain the same USB-induced artifacts that we have described so far. It appears, then, that the striations we observe in our field data are by far not unique to our own data, our own application, or our own instrumentation; they are endemic to situations in which USB keyboards have been used.

[6]To help protect user privacy, Firefox reduces timer precision by default. Users can also optionally enable fingerprint resistance, which can reduce timer precision to 100ms – obviously problematic for keystroke biometrics: `https://developer.mozilla.org/en-US/docs/Web/API/Event/timeStamp`.

7 Do the USB artifacts matter?

If the USB stack is injecting artifacts into typing data, does it matter? Clearly, from a typist's perspective, it doesn't; the pressed keys still appear on the screen in a timely manner. But when using typing times for biometric authentication purposes, or for distinguishing between legitimate and fraudulent behavior, maybe it does matter. In this section we test the idea that USB artifacts can change outcomes in an authentication regimen.

Biometric typing/keystroke data are typically used to make decisions in authentication regimes; if a user's typing rhythm matches the user's profile (which would have been obtained during an enrollment process), then the decision to authenticate the user would be made; otherwise, not. To make the authentication decision we typically use a process based on classification, matching, or similarity-based techniques to answer the question: is this person's current typing rhythm similar enough to the person's profile that we can judge the person and the profile to have arisen from the same individual? If the answer is "yes" then the user is granted permission to access system resources; otherwise, permission is denied.

To achieve this, researchers and practitioners use statistical or machine-learning techniques (e.g., classifiers) such as support vector machines, decision trees, random forests (a powerful variant of a decision tree), etc. – see, for example: [3, 10, 23]. Random forests are one of the most widely used methods in machine learning, whose predictive performance can compete with any other supervised learning algorithm. We use a random forest in this work, partly because of the aforementioned, and partly for consistency, since we have used the technique in prior work (e.g., [17]) with similar data.

To determine the extent to which USB-induced artifacts can influence programs that use keystroke data, we ask these questions:

1. does a standard classifier (e.g., random forest) get results on the USB-influenced field data that differ from its results on the lab data; and
2. when comparing artifact-free data with artifact-laden data, are there any differences in decision making outcomes?

7.1 Determining the effects of USB-induced artifacts

To demonstrate whether or not the USB-induced artifacts have any effect on decision outcomes we need to compare two data sets – one that's artifact free and one that's contaminated with artifacts. While our artifact-free lab data could be contrasted with our artifact-adulterated field data, a confounding factor would make such a comparison invalid. A confound exists when there is more than one explanation for an outcome; in this case, a change in outcome could be due to the USB quantizations, or it could be due to there being different people in the lab data than in the field data, with different typing behaviors across the data sets. To avoid this confound, we find a way to compare a single, unadulterated data set against a modified version of itself by transforming lab data into USB-quantized "field-similar" data.

In a procedure that we call "quantization" we resample the lab data (all features), and transform it so that it reflects the striated characteristics of the field data. This is implemented by the following function where t_{high} is a feature time in the high-resolution lab data set (hold time, up-down latency,

or down-down latency), t_{quant} is a feature time in the quantized lab data set, p is the polling interval, and the round function implements a banker's round-half-to-even policy:

$$t_{quant} = \text{round}\left(\frac{t_{high}}{p}\right) \times p$$

This rounding policy (see [20]) was used to avoid biasing the transformed data as would occur when rounding in the same direction, up or down, all the time; rounding in the same direction can result in a bias that grows larger and larger as more rounding operations are performed. Banker's rounding avoids this bias. As an example, 3.5 will round up to 4 and 4.5 will round down to 4. Thus we avoid biasing the mean, which can be central to some classification algorithms.

7.1.1 Establish baselines

The first step in determining the effects of USB-induced artifacts is to establish baselines for comparisons between lab and quantized, field-similar data. Using the lab data we determine the overall baseline classification accuracy of a random forest, and we create the baseline misclassification matrix of the classification outcomes. The misclassification matrix is a 100-by-100 matrix in which the element in row i, column j is the percentage of times the subject with true ID i was predicted to have ID j by the random forest. These will be compared against accuracies and misclassification matrices obtained in the experiments described below.

7.1.2 Quantize all 100 subjects

In this experiment we quantized all 100 subjects in the lab data set, in two phases: (1) quantize to 8 milliseconds; (2) quantize to 16 milliseconds. These two quantization intervals were chosen because they were the most prevalent in the field data (see Table 3, page 11).

For a given quantization level (e.g., 8 ms) we generated a misclassification matrix. That matrix was compared, cell by cell, against the cells in the aforementioned baseline matrix, seeking the cell with the greatest difference between baseline and quantized data – this cell represents the largest change across subjects' classification outcomes. We report the overall classification accuracy, the corresponding subject number, the quantization level, and the maximum cell difference in Section 8, Results.

7.1.3 Quantize just one subject – one-shot

In this experiment we determined what happens when just a single subject, out of 100, is USB-quantized; that's 1% of the data. The purpose of this experiment is to examine the changes in classification outcomes when data from only one subject are contaminated by USB artifacts. Again, this was done at both 8 and 16 millisecond intervals. Each subject was quantized, in turn, resulting in 100 classification accuracies and 100 misclassification matrices for each quantization level. For each of the two quantization levels we found the subject with the maximum difference in any cell of the misclassification matrix. As above, we report the overall classification accuracies, subject number, and the maximum cell differences in Section 8, Results.

7.1.4 Quantize two subjects – two-shot

This experiment reveals what happens when two subjects are quantized simultaneously at either 8 or 16 millisecond intervals. The purpose of this experiment is to examine the changes in classification outcomes when data from only two subjects are contaminated by USB artifacts. We could carry on, with experiments that quantize 3 or 4 or n subjects at a time, but we limit our explorations to two subjects for the sake of concision. Comparisons against the baseline were made similarly to the one-subject condition above, and similarly reported in Section 8, Results.

7.1.5 Quantize in proportion to the field data

This experiment deviates from the others in that the quantization was done in proportion to the polling rates in the original field data (see Table 3, page 11). So, one subject was quantized at 2 milliseconds, seven subjects at 4 milliseconds, 54 subjects at 8 milliseconds, and the rest in accordance with the table. To avoid obtaining a particular result by coincidental assignment of subjects to quantization levels, the assignments were done by random draw, and repeated ten times, with the results averaged.

In the next section we will present the results regarding the several overall accuracies and misclassification matrices that were produced from these experiments.

8 Results

In the previous section we described our experiments in quantization and the concomitant generation of their overall classification accuracies and misclassification matrices. In this section we report those results.

In addition to (modest) changes in overall classification accuracies due to USB-induced artifacts, there were changes in the misclassification matrices generated in every experiment. Here we show the major changes on the diagonal of the misclassification matrix – that's the cell whose value shows the number of (or the percentage of) times that a subject was classified as him/herself. We follow the same structure as we did earlier in Section 7.

8.0.1 Establish baselines

The average overall classification accuracy for unquantized lab data was 90.57% (std = 0.0007), based on ten runs of the random forest classifier. Because the differences among the ten runs were so small, we decided to use the first run for our baseline; for that run the overall classification accuracy was 90.55%. The baseline misclassification matrix isn't shown here because of its 100 x 100 size, although excerpts will be shown later in this section. This is the baseline against which classification results for the quantized, field-similar data will be compared.

8.0.2 Quantize all 100 subjects

The classification accuracy of data – following the same procedure as above – quantized at 8 milliseconds was 90.37%; at 16 milliseconds it was 89.46%. These compare with 90.55% for the unquantized lab data, differences of meager consequence.

8.0.3 Quantize just one subject – one-shot

In this experiment we determined what happens when just a single subject (we call this a one-shot quantization), out of 100, is USB-quantized; that's 1% of the data. Again, this was done at both 8 millisecond and 16 millisecond intervals. The overall classification accuracies were 90.69% and 90.85% for 8 and 16 milliseconds, respectively. The differences between these and the baseline of 90.55% are of practically no consequence.

At this juncture it is appropriate to draw attention to changes in the misclassification matrix generated by the random forest. Table 4 shows excerpts from two misclassification matrices, one from the baseline (top) and the other from the 16 millisecond one-shot quantization just reported. Comparing the two matrices, there are 1030 matrix cells that differ. The largest of these differences is reflected in the behavior of subject 032, whose diagonal value is 0.62 (in red) in the baseline condition, and 0.78 in the one-shot condition – a difference of 16 percentage points. That's enough of a difference to cause a change in a decision, from legitimate to fraudulent, or vice versa, depending on what the decision threshold is. This is a serious difference, caused by the artifacts induced by the USB stack. We will see more such digressions in the two cases below, two-shot and proportional.

	s029	s030	s032	s034	s035
s029	0.805	0	0	0	0
s030	0	0.975	0	0	0
s032	0	0.005	0.62	0.005	0
s034	0	0	0	0.825	0
s035	0	0	0	0	0.895

Misclassification matrix excerpt, baseline, raw lab data.

	s029	s030	s032	s034	s035
s029	0.805	0	0	0	0
s030	0	0.965	0	0	0
s032	0	0.01	0.78	0.01	0
s034	0	0	0	0.81	0
s035	0	0	0	0	0.895

Misclassification matrix excerpt, one-shot, s032, 16ms quantized data

Table 4: Misclassification matrices, excerpts. Upper: baseline. Lower: 1-shot experiment. Red cells, 16 point difference.

Quant level	Classif accuracy	Cells changed	Most Egregious	Diagonal Change	Percentage Points
(a) All data					
–	90.55	—	—	—	—
8	90.37	1073	s079	.785/.830	4.5
16	89.46	1190	s034	.825/.765	6
(b) One-shot (8ms: s079 / 16ms: s032)					
8	90.69	1038	s079	.785/.875	9
16	90.85	1030	s032	.620/.780	16
(c) Two-shot (8ms: s007 & s111 / 16ms: s018 & s032)					
8	90.52	1038	s007	.755/.870	11.5
16	90.87	1058	s032	.620/.810	19
(d) Proportional (averages among 10 runs)					
–	91.48	1084	Various	.802/.883	8.15

Table 5: Effects of quantization; 4 cases. Example, Section b: when one subject (out of 100, hence one-shot) was quantized at 16 ms, there was a 16 percentage point change on subject s032's diagonal.

Table 5-b shows the same result: a 16-millisecond quantization of a single subject (s032) corresponds to a shift of 16 percentage points (from 62% correct classification before quantization to 78% after) in the value of the s032 diagonal. Similarly, an 8-millisecond quantization of a single subject (s079) corresponds to a 9-point shift.

8.0.4 Quantize two subjects – two shot

The overall classification accuracies were 90.52% and 90.87% for 8 and 16 milliseconds, respectively. Again, the differences between these and the baseline of 90.57% are of practically no consequence; and neither do these accuracies vary significantly from the accuracies revealed in the experiments described above. More concerning, as Table 5-c shows, is that when two subjects are quantized at 16 milliseconds, the shift on the diagonal for subject s032 is 19 percentage points, a shift that's even worse than for the one-shot case.

8.0.5 Quantize in proportion to the field data

This experiment deviates from the others in that the quantization was done in proportion to the polling rates in the original field data (see Table 3, page 11). To illustrate, one subject was quantized at 2 milliseconds, seven subjects at 4 milliseconds, 54 subjects at 8 milliseconds, and the rest in accordance with the table. To avoid obtaining a particular result by coincidental assignment of subjects to quantization levels, the assignments were done by random draw, and this was repeated ten times, with the results averaged. The overall average classification accuracy was

91.48%, again not significantly different from the other overall classification accuracies in this suite of experiments.

Table 5-d shows that on average there were 1084 cell-value changes, and on average 8.15 percentage point shifts in values on the diagonal; still enough to change decision outcomes.

9 Discussion and summary

9.1 Research questions

The study addressed these questions: (1) What distinguishes lab vs field data? (2) If distinctions exist, do they matter?

9.1.1 What distinguishes lab vs field data?

We explored two data sets (lab and field) to discover and elucidate any differences between them. We used standard techniques from exploratory data analysis [32]. We discovered that in the field data the recorded keystroke timestamps were not faithful to the original signals from the keyboard because of the way that USB keyboards work (e.g., polling) in conjunction with a computer operating system. (It is worth noting that this would be true for any operating system, not just Windows.) The consequence of this infidelity was that keystroke timings in the field data, rather than reflecting genuine keystroke timings, were centered around USB polling intervals, the most common of which was 8 milliseconds (cf. Table 3). Keystroke timings in the lab data showed no such distortions; see Figure 2, comparing USB-based striations vs distortion-free lab data.

9.1.2 Do the lab/field differences matter?

The effect of the USB-induced timing distortions was shown in changes to classification results. We ran each data set, lab and field, through a random-forest classifier and got different results for each data set; but those differences, unfortunately confounded, were due to each data set having originated from a different subject pool, not from timing artifacts.

To determine the impact of the timing artifacts we compared the lab data against field-similar (quantized) lab data. Field-similar data are lab data transformed to resemble the artifact-ridden field data. Therefore, we are comparing the exact same subject pool with itself, but with USB timing characteristics. We did this in several stages, and at different quantization levels: 8 ms, 16 ms, and proportional to the polling in the original field data (cf. Table 3). We found that even if only 1 or 2 percent of the data are infected with USB-induced artifacts, decision outcomes can change by as much as 19 percentage points, which is unacceptable in most situations. We also tested other classifiers (KNN and SVM), obtaining similar results; hence the particulars of any individual classifiers were not at issue.

9.2 Why this happened

We reported only a few examples out of a vast collection of other people's data, all of which contained USB-induced artifacts. Why have so many data sets been collected with USB timing disparities in them? Perhaps it's due to architectural naïveté. Although most practitioners in keystroke biometrics are computer scientists or machine learning specialists, perhaps their knowledge of computer architecture was too limited to anticipate timing issues. In our own work, we knew about extant timing difficulties, but didn't know their cause; hence, in our lab-based studies, we used a custom timer to avoid them. When we moved to the real world, we were forced to understand their origins and consequences.

9.3 Now what?

The dominant modern keyboard type is USB. All USB keyboards follow a polling protocol, so there is no way to avoid the quantization shown in this work, and exemplified in Figures 2 and 3. Maybe the quantization problem itself could be mitigated by a workaround if all keyboards polled at the same rate, but they don't. Moreover, as shown in Section 5.1 on unruly data, distortions in the data go beyond what can be explained by USB polling protocols. All such artifacts can influence outcomes in statistically-based decision algorithms.

It might seem that there are no solutions, but that depends on how keystroke biometrics are being used. In a closed environment, such as a corporation or a government department, keyboards can be modified to avoid or compensate for the USB problem. We did this in our own lab; the parts cost less than five dollars (for one device), so in quantity the modification would be cheap. On a broader scale, we can hardly expect everybody to modify their keyboards, at least in the short term. However, because the USB protocol's inherent flexibility (as evidenced by its backwards-compatible evolution from 1.1 to USB 3.0) allows for additions such as biometrics into HID data without disruption or change to existing equipment (e.g., keyboards or PCs), it's possible that the keyboard industry could respond to demand for a faster-polling biometric device.

For today's researchers, the USB problem might seem insurmountable unless data are gathered in a controlled environment, or unless a future innovation is found to overcome the unpredictable effects of USB polling artifacts. Of course there are applications other than authentication where the signal rises so far above the noise that USB polling will have only negligible effects.

To some, our paper may suggest a bleak future for keystroke biometrics. However, we see the issues identified here as challenges to researchers: to adapt to the demands of the field and to produce high-quality data regardless of source.

A key recommendation is that researchers should screen their data, preferably as it comes in, but at least before engaging in any analysis. As we showed in Section 6, other people's data, all kinds of things can go wrong, and it's best to discover them before putting the data to use.

10 Limitations

We only used the Windows operating system for data collection, but the operating system likely doesn't matter; what matters is the keyboard. A real-time computer system might do better, coupled with a fast keyboard, but we are addressing commodity systems, not specialized systems.

11 Lessons learned

The complexity of modern operating systems and communication protocols makes it hard (impossible?) to obtain reliable and consistently timed data. Given that, and the issues discussed in this paper, it would seem imperative to screen data before using it. This is not merely to check for USB polling, but also to check for what we called "unruly" data – data that are so outside expectations that they are almost sure to be deleterious for the data application.

12 Conclusion

We discovered that USB keyboards inject timing artifacts into keystroke biometric data that are unpredictable and detrimental to a statistical decision algorithm (e.g., random forest). In a systematic exploration of other people's data, we found a range of damaging and unexplained artifacts, with the appearance of being USB-induced, in every single data set that we examined. Such artifacts can change the basis of a decision by nearly 20 percentage points, possibly reversing a claim of legitimacy into an accusation of fraudulence, or denying a claim of identity and preventing the admission of a legitimate user to sensitive resources. Researchers and practitioners would do well to screen their data before employing them in decision-making algorithms, especially in critical applications like international banking and finance, noting the European Banking Authority's endorsement of keystroke biometrics as a strong authentication mechanism.

References

[1] Salil P. Banerjee and Damon L. Woodard. Biometric authentication and identification using keystroke dynamics: A survey. *Journal of Pattern Recognition Research*, 7(1):116–139, 2012.

[2] Francesco Bergadano, Danielle Gunetti, and Claudia Picardi. User authentication through keystroke dynamics. *ACM Transactions on Information and System Security (ACM TISSEC)*, 5(4):367–397, November 2002.

[3] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, October 2001.

[4] William Lowe Bryan and Noble Harter. Studies in the physiology and psychology of the telegraphic language. *Psychological Review*, 4(1):27–53, January 1897.

[5] Compaq Computer Corporation, Hewlett-Packard Company, Intel Corporation, Lucent Technologies Inc., Microsoft Corporation, NEC Corporation, and Koninklijke Philips Electronics N. V. *Universal Serial Bus Specification, Revision 2.0*, 27 April 2000. URL: `https://www.usb.org/document-library/usb-20-specification`.

[6] Paul-Olivier Dehaye. Coursera and keystroke biometrics. Retrieved from `https://medium.com/personaldata-io/coursera-and-keystroke-biometrics-550762f2f61b`, 12 February 2016.

[7] Paul-Olivier Dehaye. Coursera_Safe_Harbor_requests. Retrieved from `https://github.com/pdehaye/Coursera_Safe_Harbor_requests/blob/master/typing_sample.csv`, 12 February 2016.

[8] Vivek Dhakal, Anna Maria Feit, Per Ola Kristensson, and Antti Oulasvirta. Observations on typing from 136 million keystrokes. In *CHI 2018: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 646:1–646:12, New York, NY, 21-26 April 2018. ACM.

[9] Salima Douhou and Jan R. Magnus. The reliability of user authentication through keystroke dynamics. *Statistica Neerlandica*, 63(4):432–449, November 2009.

[10] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley and Sons, Inc., New York, second edition, 2001.

[11] European Banking Authority (EBA). Opinion of the European Banking Authority on the elements of strong customer authentication under PSD2. Paris, France; 21 June 2019.

[12] Romain Giot, Bernadette Dorizzi, and Christophe Rosenberger. A review on the public benchmark databases for static keystroke dynamics. *Computers & Security*, 55(4):46–61, November 2015.

[13] Romain Giot, Mohamad El-Abed, and Christophe Rosenberger. Web-based benchmark for keystroke dynamics biometric systems: A statistical analysis. In George A. Tsihrintzis, Jeng-Shyang Pan, Hsiang-Cheh Huang, Maria Virvou, and Lakhmi C. Jain, editors, *2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP 2012)*, pages 11–15, Los Alamitos, CA, 18-20 July 2012. IEEE Computer Society.

[14] IBM. IBM Personal System/2 Hardware Interface Technical Reference. First Edition (May 1988). Retrieved from `http://bitsavers.trailing-edge.com/pdf/ibm/pc/at/1502494_PC_AT_Technical_Reference_Mar84.pdf`, May 1988.

[15] Pilsung Kang, Sunghoon Park, Seong-seob Hwang, Lee Hyoung-joo, and Sungzoon Cho. Improvement of keystroke data quality through artificial rhythms and cues. *Computers & Security*, 27(1-2):3–11, March 2008.

[16] Kevin Killourhy and Roy Maxion. The effect of clock resolution on keystroke dynamics. In Richard Lippmann, Engin Kirda, and Ari Trachtenberg, editors, *11th International Symposium on Recent Advances in Intrusion Detection (RAID 2008)*, volume 5230 of *Lecture Notes in Computer Science (LNCS)*, pages 331–350, Berlin, 15-17 September 2008. Springer Verlag.

[17] Kevin S. Killourhy and Roy A. Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN-09)*, pages 125–134, Los Alamitos, California, 29 June - 02 July 2009. IEEE Computer Society Press. Estoril, Lisbon, Portugal.

[18] Yilin Li, Baochang Zhang, Yao Cao, Sanqiang Zhao, Yongsheng Gao, and Jianzhuang Liu. Study on the BeiHang keystroke dynamics database. In *Proceedings of the 2011 International Joint Conference on Biometrics (IJCB-11)*, pages 1–5, Washington, DC, 10-13 October 2011. IEEE Computer Society.

[19] Juan Liu, Baochang Zhang, Haoran Zeng, Linlin Shen, Jianzhuang Liu, and Jason Zhao. The BeiHang keystroke dynamics systems, databases and baselines. *Neurocomputing*, 144:271 – 281, 2014.

[20] Clive Maxfield. An introduction to different rounding algorithms. Electronic Engineering Times, 04 January 2006.

[21] Roy A. Maxion and Kevin S. Killourhy. Keystroke biometrics with number-pad input. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN-10)*, pages 201–210, Los Alamitos, California, 28 June - 01 July 2010. IEEE Computer Society Press. Chicago, Illinois.

[22] Microsoft Corporation. Acquiring high-resolution time stamps. Windows Dev Center, 31 May 2018. 31 May: (`https://docs.microsoft.com/en-us/windows/win32/sysinfo/acquiring-high-resolution-time-stamps`).

[23] Tom M. Mitchell, editor. *Machine Learning*. McGraw-Hill, Boston, 1997.

[24] Aythami Morales, Julian Fierrez, Ruben Tolosana, Javier Ortega-Garcia, Javier Galbally, Marta Gomez-Barrero, André Anjos, and Sébastian Marcel. Keystroke biometrics ongoing competition. *IEEE Access*, 4:7736–7746, November 2016.

[25] Mohammad S. Obaidat, P. Venkata Krishna, V. Saritha, and Shubham Agarwal. Advances in key stroke dynamics-based security schemes. In Mohammad S. Obaidat, Issa Traore, and Isaac Woungang, editors, *Biometric-Based Physical and Cybersecurity Systems*, pages 165–187. Springer International Publishing, Cham, Switzerland, 2019.

[26] M.S. Obaidat and Balqies Sadoun. A simulation evaluation study of neural network techniques to computer user identification. *Information Sciences*, 102(1-4):239–258, November 1997.

[27] Narainsamy Pavaday, Sunjiv Soyjaudah, and Shrikaant Nugessur. Investigating & improving the reliability and repeatability of keystroke dynamics timers. *International Journal of Network Security & Its Applications (IJNSA)*, 2(3):70–85, July 2010.

[28] Alen Peacock, Xian Ke, and Matthew Wilkerson. Typing patterns: A key to user identification. *IEEE Security and Privacy*, 2(5):40–47, September/October 2004.

[29] Mariusz Rybnik, Piotr Panasiuk, Khalid Saeed, and Marcin Rogowski. Advances in the keystroke dynamics: The practical impact of database quality. In Agostino Cortesi, Nabendu Chaki, Khalid Saeed, and Sławomir Wierzchoń, editors, *Computer Information Systems and Industrial Management. (CISIM 2012)*, volume 7564 of *Lecture Notes in Computer Science*, pages 203–214, Berlin, Heidelberg, 26-28 September 2012. IFIP, Springer.

[30] Yan Sun, Hayreddin Ceker, and Shambhu Upadhyaya. Shared keystroke dataset for continuous authentication. In *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 4-7 December 2016. Abu Dhabi, United Arab Emirates.

[31] Pin Shen Teh, Andrew Beng Jin Teoh, and Shigang Yue. A survey of keystroke dynamics biometrics. *The Scientific World Journal*, 2013, August 2013. Article ID 408280, 24 pages.

[32] John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, Lebanon, Indiana, 1977.

[33] Yu Zhong and Yunbin Deng. A survey on keystroke dynamics biometrics: Approaches, advances, and evaluations. In Yu Zhong and Yunbin Deng, editors, *Recent Advances in User Authentication Using Keystroke Dynamics Biometrics*, pages 1–22. Science Gate Publishing, Xanthi, Greece, 2015.

Appendix: Reproducibility

Information for those wishing to reproduce our study.

- Hardware platform (lab data collection)
  - IBM ThinkPad X60s (model 1702-4EU); 1.5 GB RAM
  - Apple M9034LL/A USB external keyboard
- Software platform (lab data acquisition)
  - Windows XP Professional (Service Pack 2)
  - All services and networking disabled
- Hardware platform (all data analysis)
  - Model Name: MacBook Air
  - Model Identifier: MacBookAir6,2
  - Processor Name: Intel Core i5
  - Processor Speed: 1.4 GHz
  - Number of Processors: 1
  - Total Number of Cores: 2
  - L2 Cache (per Core): 256 KB
  - L3 Cache: 3 MB
  - Memory: 4 GB RAM
  - Boot ROM Version: MBA61.0099.B53
  - SMC Version (system): 2.13f15
- Software platform (data analysis)
  - Operating System: macOS Sierra version number 10.12.6 (Build number: 16G29)
  - Python: Version 3.7.1, 20 October 2018
  - matplotlib version 3.0.2 (for plotting)
  - numpy version 1.18.4 (for random seeds)
  - pandas version 0.23.4 (for working with csv data)
  - scikit-learn version 0.21.3 (for the random forest)
  - scipy version 1.4.1 (for stats like iqr and mode)
  - Random forest classifier (Scikit-learn)
    - The training data were split from the testing data by taking a random draw of 25 repetitions from each session. The random draw was initialized using a seed of 0.
    - The random forest was run with the random_state parameter set to 0.
- Script: regenerates all results
- Data: Contact authors.