SoK: Keylogging Side Channels

John V. Monaco
U.S. Army Research Laboratory
Aberdeen Proving Ground, MD, USA

***Abstract*—The first keylogging side channel attack was discovered over 50 years ago when Bell Laboratory researchers noticed an electromagnetic spike emanating from a Bell 131-B2 teletype terminal. This spike, emitted upon each key press, enabled up to 75% of plaintext communications to be recovered in field conditions. Since then, keylogging attacks have come to leverage side channels emanating from the user's finger and hand movements, countless keyboard electromagnetic and acoustic emanations, microarchitectural attacks on the host computer, and encrypted network traffic. These attacks can each be characterized by the type of information the side channel leaks: a spatial side channel reveals physical key locations or the similarity between key pairs, and a temporal side channel leverages key press and release timings. We define and evaluate the performance of idealized spatial and temporal keylogging side channels and find that, under the assumption of typing English words, nontrivial information gains can be achieved even in the presence of substantial measurement error. For temporal side channels, we find that the information gained by different temporal features strongly correlates to typing speed and style. Finally, to help drive future research, we review the current state-of-the-art keylogging side channel attacks and discuss some of the mitigation techniques that can be applied.**

I. Introduction

In 1984, sixteen electromechanical bugs were discovered inside IBM Selectric II and Selectric III typewriters at the U.S. Embassy in Moscow and U.S. Consulate in Leningrad [1]. The bugs, developed by the Soviet Union (USSR), contained 6 magnetometers that sensed upon each keystroke the actuation of 6 individual levers in the Selectric typewriter. The unique combination of actuated levers enabled the Selectric Bug to determine which key had been pressed, ignoring Shift, Space, and some other non-letter keys. Up to 8 key presses were stored in an 8×4-bit core memory, and when this became full the memory contents were transmitted via radio bursts to a nearby listening post. The Soviets developed at least 5 versions of the Selectric Bug which enabled them to log the keystrokes of their adversaries for more than 8 years [1].

Keystroke logging, or *keylogging*, is the practice of recording the keys a person types on a keyboard. This can often be accomplished by means of a *side channel attack*, whereby an unintended information source is leveraged. The Selectric Bug is one of the first such hardware keyloggers discovered in the wild and represents a milestone in surveillance technology. It is perhaps one of the most successfully executed attacks on keyboard input known to the public and helped raise awareness to the real possibility of covertly logging keystrokes. The idea itself of using a side channel attack to log keystrokes dates back over 50 years to the well-known TEMPEST Program, prompted by the discovery of Bell Laboratory researchers that emanating electromagnetic spikes from a teletype terminal could be used to effortlessly decode secure communications [2], [3]. In the years to come, keylogging side channel attacks would evolve from typewriter to computer keyboard and grow to cover a broad spectrum of vulnerabilities. Likewise, relatively sophisticated keylogging defenses have been developed to address a diverse set of attack scenarios.

Fig. 1. Selectric Bug, one of the first keylogging side channel attacks discovered in the wild. Reprinted with permission from [4] and on display in the National Cryptologic Museum.

There currently exists a number of modalities with which keystrokes can be detected and identified, including WiFi signal distortion [5], electromagnetic (EM) emanations [6], CPU cache usage [7], and network traffic patterns [8]. Conversely, side channel mitigation can in some cases be achieved by low-cost countermeasures that aim to reduce emanations [9] and mask typing behavior [10]. Each attack relies on a side channel that emanates from either the user, the host computer, the keyboard, or the network, respectively. And while some attacks utilize a side channel that leaks only temporal information, such as the timing of key press and release events, others leverage spatial information to reveal the location of and physical distance between keys on the keyboard.

This work aims to establish a framework within which keylogging side channel attacks and defenses are described and evaluated. Keylogging performance metrics are defined, and the effects of spatial measurement error and typing speed of the victim are determined for idealized spatial and temporal side channels. To help drive future research, we attempt to consolidate much of the core knowledge in keyboard mechanics and review the current state-of-the-art keylogging side channel attacks and defensive mitigation techniques.

The rest of this article is organized as follows. Section II reviews the basic operation and communication protocols of the keyboard, providing the basis for many side channel attacks. Timing issues are also considered, including sources of delay and the effects of process scheduling. Section III formally defines the problem of keylogging with respective performance metrics and introduces the concept of idealized spatial and temporal side channels. Specific keylogging attacks and defenses are characterized in Sections IV and V, respectively. Section VI discusses privacy concerns and identifies directions for future research. Section VII concludes and the Appendix provides a summary of public keystroke datasets.

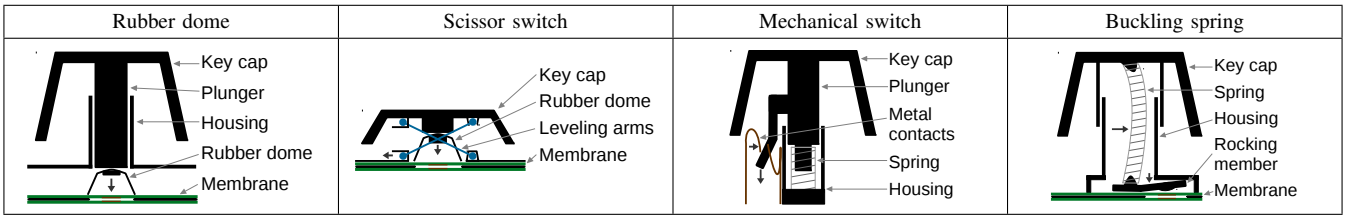

Fig. 2. Common keyboard switch designs.

II. Background

There exist a great variety of keyboard[1] types that differ in the way keys are actuated, sensed, encoded, and reported to the host. In this section, some of the most widely used components and communication protocols are reviewed.

A. Keyboard Layout

The modern computer keyboard is the heir to the first commercially successful typewriter invented by Christopher Latham Sholes [12]. Today, a distinction can be made between the *logical layout*, which defines a mapping of key identifiers to physical key locations (e.g., QWERTY vs DVORAK), and the *physical layout*, regarding the shape and arrangement of keys on the keyboard. The dimension and placement of keys is often specified by the relative *unit* measurement, which is agnostic to the physical dimensions of the keyboard: 1 unit (u) is the length and width of the space occupied by square alphanumeric keys, and on most keyboards, the horizontal center-to-center key distance. The dimensions of other keys can be specified in terms of keyboard units: e.g., on the ANSI 101 physical layout, Tab is 1×1.5 u, Caps Lock is 1×1.75 u, and LShift is 1×2.25 u [13]. The physical distance tied to 1 u varies depending on specific keyboard, although the ISO standard specifies 1 u$\equiv$19$\pm$1 mm for full-sized keyboards [14].

B. Sensing and Actuation

The *sensing method* refers to the way in which a keystroke is physically sensed [15]. Sensing can be performed through conductance, capacitance, photo-conductance, or even Hall effect, a method intended for applications that require extreme reliability, such as aircraft cockpits [16]. Most commodity keyboards since the mid 1990's utilize a flexible conductive membrane with printed circuits to detect keys. The membrane is comprised of three layered plastic sheets with circuit traces on the top and bottom sheets. The middle sheet contains cavities at each key location so that when pressure is applied to the area above the cavity the circuit is completed.

The *switch* refers to the actuation mechanism that interfaces the sensor, also providing tactile and auditory feedback to the user. Among the common switch types, shown in Figure 2, are the rubber dome, scissor switch, mechanical switch, and buckling spring. Most commodity keyboards consist of a *rubber dome* switch on top of a membrane layer [17]. The key cap sits on top of a plunger which, when pushed, causes a rubber dome to apply pressure to the membrane and complete the circuit. The typical rubber dome switch provides 3.5–4 mm of travel and requires 50–70 cN of force to actuate, although the latter varies based on specific manufacturing process [18].

Stabilized, or *scissor* switch, keys are common in laptops and low-profile "chiclet" keyboards [19], [20]. Scissor switch keys are characterized by a set of leveling arms that stabilize the key cap upon actuation, traveling a distance of up to 2 mm. Although they have a lower profile than standard rubber dome keyboards, the actuation force is comparable [18].

Unlike rubber dome and scissor switch keyboards, metal contact switch, or *mechanical* switch, keyboards sense keystrokes through the conductance of metal contacts as opposed to an underlying membrane. Most mechanical switches operate by *negative action*, whereby the switch assembly displaces an object that holds metal contacts apart; when the object is removed, the metal contacts close on their own accord. This design ensures more consistent closing behavior than *positive action*, whereby the switch assembly brings the metal contacts to together.

Buckling spring switches, which emit a characteristic sound, predate the modern rubber dome-over-membrane design [21]. Popularized by IBM in the 1970's and 1980's, the buckling spring switch operates by compressing a spring to the point of catastrophic buckling upon which it pivots a rocking member that actuates the sensor below, such as a membrane.

C. Matrix Scanning

A major function of the *microcontroller* is to detect any pressed keys. To avoid having a dedicated line for each key, this is usually performed by multiplexing a matrix circuit design such that the intersection of each column (scanning line) and row (feedback line) forms a switch for each key [22]. The microcontroller continuously scans the matrix, pulsing each column for a short duration (up to 3 µs) at a rate referred to as the *matrix scan rate*. Any key pressed along the pulsed column completes a circuit which is detected at the corresponding row. In this way, a key press can only be detected when the microcontroller pulses the appropriate column. Although cost efficient, this design can lead to *ghosting*, or the erroneous detection of a fourth key when three adjacent keys in the matrix are pressed. Similarly, *masking* occurs when three adjacent keys in the matrix are simultaneously pressed and a fourth key cannot be detected. Key arrangement within the matrix, and some redundancy, can mitigate these effects. The matrix of a typical 101-key keyboard has up to 24 columns and 8 rows, which can support up to 192 unique keys [22].

[1]The focus of this work is on full-size hardware keyboards and not "soft" touchscreen keyboards. For an example of an attack on the latter, see [11].

D. Communication Protocols

There are predominately two ways in which keyboards communicate with the host computer, PS/2 and USB, differing in the way keystrokes are both encoded and transmitted.

1) PS/2: Each physical key is assigned a unique scancode which is transmitted to the host by the keyboard in an 11-bit frame comprised of: a start bit (always 0), 8 bits for the scancode (least significant bit first), odd parity check bit, and a stop bit. For each bit, the keyboard pulls down the clock signal, which must be in the range 10–16.7 kHz and is provided by the keyboard itself, and then sends the bit over the data line, i.e., data should be sent from the keyboard to the host on the falling edge of the clock signal [23].

2) USB: Unlike PS/2, USB keyboards are passive in nature [24]. The universal host controller interface (UHCI) periodically issues a query to the keyboard and the keyboard responds with its current state indicating which keys are pressed [25]. The polling interval of the UHCI determines the maximum rate that the keyboard can report any changes in state, which ranges anywhere from 100 Hz to 1000 Hz (125 Hz is a common default). Most USB keyboards implement USB 1.x since they require relatively low bandwidth. Even low speed USB devices, which use a 6 MHz clock and non return to zero inverted (NRZI) encoding, can achieve much higher transmission rates than PS/2.

The USB keyboard state is an 8-byte packet that contains a modifier key mask (byte 0), a reserved byte (byte 1), and up to 6 keys that are currently pressed (bytes 2-7). There is no way to send a "release" event as the keyboard can only report that some keys are in a pressed state. Because of this, the host computer must maintain its own state of the keyboard, inferring key releases when necessary. The order of keys in the packet does not matter, and due to the size of the packet, USB keyboards are limited to 8 modifier keys and 6 non-modifier keys (commonly referred to as 6-key-rollover, or 6KRO).

3) Keyboard Interrupts: After the key press or release event is encoded by the keyboard and transmitted to the host, a hardware interrupt is raised on the host. The hardware interrupt flags the CPU to respond to the keyboard event, which typically involves reading the scancode (PS/2) or key identifier (USB) from the keyboard buffer. Followup processing, such as making the event available to shared libraries and user applications, may be performed later since the kernel must respond as quickly as possible to the hardware interrupt. Until the keyboard buffer is cleared and the interrupt is acknowledged by the host, no further keystrokes can be received.

E. Timing: Sources of Delay and Variability

Due to the physical structure of the keyboard, a key press is always followed by a key release, and together they form a full keystroke[2]. From the perspective of the host computer, a keystroke is a 3-tuple with the physical key identifier k, the press time t^P, and the release time t^R.

A class of keylogging techniques that exploit temporal side channels, described in Section III-C, rely on the precise timing of keyboard events. The temporal resolution and precision[3] of each keystroke depends on the sampling rate of the sensor, where in the processing pipeline it was detected, the speed and scheduling policy of the host computer, and the keyboard itself. There are several sources of delay and timing variability that can affect both the resolution and precision with which keystrokes are measured. These include:

1) Physical Delay: The time from physical contact between the user's finger and key cap to the point of actuation depends on characteristics of the switching mechanism in the keyboard, such as travel distance, feedback profile, and actuation force. As described in Section II-B, these properties vary between keyboard types. From the perspective of the host, the physical delay cannot be measured since a keystroke is sensed only at the point of actuation. However, an external sensor in proximity to the user, such as a microphone or motion sensor, can track the user's hand movements and consider the physical delay if necessary (see Figure 8).

2) Matrix Scan Rate: The rate at which the microcontroller pulses the scanning lines in the keyboard matrix varies between keyboard models, ranging anywhere from 100 Hz to 400 Hz [26]. If a key press or release occurs just after the corresponding column was pulsed, it will have to wait until the next pulse arrives. This introduces a maximum delay of $d_{sc} = 1000/f_{sc}$ ms with, assuming a uniform distribution of delays, mean $d_{sc}/2$ ms and standard deviation $\sqrt{d_{sc}^2/12}$ ms, where f_{sc} is the scanning frequency.

3) Debouncing: When a key is pressed, it closes a switch in underlying the circuit. This does not occur instantaneously, but rather takes time for the switch to reach a stable closed state, commonly referred to as *switch bouncing*. In order to not generate spurious keystrokes, debouncing must be applied. Many keyboards have a ≤ 5 ms debounce timeout [27], [28].

4) Encoding: Once the keyboard microcontroller determines that a key has been pressed, it enters a subroutine to convert the action into a digital signal for transmission to the host. This process interrupts the matrix scanning routine after the last scanning column that was pulsed, causing a small delay. Some side channels make explicit use of this delay to infer the column along which a key was pressed, narrowing down the possibilities to a much smaller subset of keys [6].

5) Polling Rate (USB): USB keyboards have an additional source of latency that arises from the USB polling rate. Since USB keyboards are passive in nature, they must wait for the UHCI to query their state before responding with any pressed keys. Similar to the matrix scan rate, the USB polling rate introduces a maximum delay of $d_{po} = 1000/f_{po}$ ms, where f_{po} is the USB polling frequency. PS/2 keyboards are interrupt-based, therefore do not suffer from polling delays.

6) Process Scheduling: Once the scancode reaches the host computer, a hardware interrupt is raised. The kernel

[2]This is in contrast to touchscreen keyboards which allow a key to be pressed without subsequently being released by dragging the pointing device.

[3]Resolution is the degree to which a measurement can be made and precision is the degree to which a measurement can be repeated.

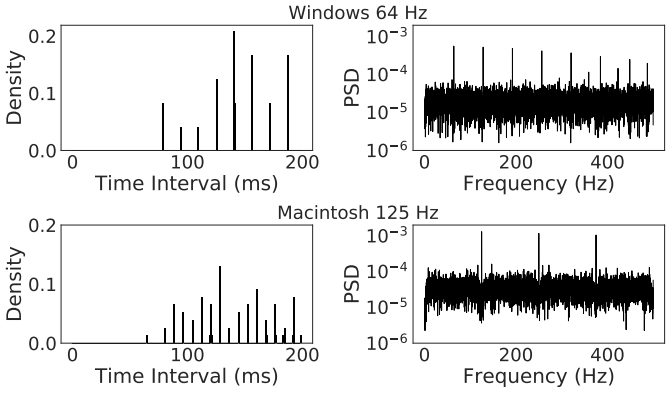

Fig. 3. Key press time intervals reveal system timer resolution (top, Windows 64 Hz) and USB polling rate (bottom, Mac OS X 125 Hz). The timer has a binning effect on the time intervals (left), resulting in peaks at harmonics of the fundamental frequency in the power spectral density (PSD, right).

must retrieve the scancode and acknowledge the interrupt, which also takes time, adding an additional constant delay. However, *when* the interrupt is actually handled depends on the scheduling policy of the kernel. This is largely determined by the scheduling clock *tick*, which specifies the time resolution with which the scheduler advances. As keystroke events are only handled on each timer interrupt, from the perspective of an application running on the host, keystroke timings will generally align to some multiple of the scheduling clock tick.

The effects of process scheduling and USB polling are apparent when keystroke events are detected on or downstream from the host, such as with CPU load [29] and network traffic [8]. Figure 3 shows the histogram and power spectral density (PSD) of key press time intervals recorded in a web browser on several different platforms. The system timer has a binning effect on the time intervals, which are tightly clustered around multiples of the scheduling clock tick. Peaks in the PSD correspond to harmonics of the fundamental timer frequency, which also reveals the scheduling clock tick. These PSD signatures could be used to perform coarse-grained host identification: 64 Hz, or a 15.625 ms tick, is characteristic of the Windows family [30] and 100 Hz of Mac OS X [31]. The Linux kernel has traditionally provided the option of 100, 250, 300, and 1000 Hz ticks through the CONFIG_HZ Kconfig parameter. On newer kernels, CONFIG_NO_HZ is set by default, enabling a "tickless" system more suitable for multimedia applications and power-constrained devices [32].

III. Side Channels

A keylogging side channel attack involves leveraging an unintended information source to determine which keyboard keys were pressed. By this definition, a device or program that senses keystrokes by directly intercepting the PS/2 or USB signal [33], registering a system hook on the host computer [34], or querying the keyboard state table [35], are not side channel attacks since these methods operate through a channel *intended* to provide information about the keyboard state. An *unintended* information source is an information source outside of the keyboard communication pipeline described in Section II-D. Such a side channel may utilize either a hardware or software sensor, e.g., a microphone to capture acoustic emanations or a program that runs on the host to measure CPU load. Keylogging side channel modalities can broadly be characterized as being either spatial or temporal: *spatial side channels* use spatial information to reveal physical key locations on the keyboard, and *temporal side channels* use keystroke timings to determine individual keys or key pairs.

A. Metrics

Keylogging is a two-step process comprised of two distinct problems: *keystroke detection* and *key identification*. Keystroke detection is the act of detecting that a keystroke has occurred at some point in time, specifically in determining t^P and/or t^R. Key identification is the act of determining k, the physical key that was pressed, given that a keystroke has been detected. While some metrics, such as Damerau–Levenshtein edit distance [36], can simultaneously capture the performance of both tasks, here we evaluate each task separately.

1) Keystroke Detection: Before a physical key is identified, the presence of a keystroke must be established. If time is sliced into successive windows of equal size, the problem of keystroke detection amounts to deciding whether each window contains a keystroke. As a binary classification problem, performance can be measured by standard metrics. The true positive rate (TPR) is the rate at which keystrokes are correctly detected and the true negative rate (TNR) is the rate at which time windows that don't contain a keystroke are correctly labeled as such. Too many false negatives will lead to sparse acquisition and provides little information, and too many false positives will obfuscate the true keystrokes.

While perfect keystroke detection has TPR=TNR=1, there is typically a tradeoff between TPR and TNR. With an acoustic side channel, keystroke detection may be performed using an energy threshold within a sliding window [37]. A threshold too low will generate many spurious results, and a threshold too high will fail to capture most true keystrokes. Similar tradeoffs exist for WiFi signal distortion [5], memory access footprints [7], and CPU load measurements [29].

2) Key Identification: Unlike keystroke detection, key identification is a multiclass classification problem. Given a keystroke has been detected, the problem of key identification is to determine which physical key on the keyboard was pressed. In case of a keystroke sequence, this can also be performed at the word level. There are $|\mathcal{K}|^n$ ways to label a sequence of n unknown keystrokes, where $|\mathcal{K}|$ is the cardinality of the set of all possible keys $\mathcal{K}$.

The intrinsic entropy of a single keystroke with unknown key k depends on the probability $P[k]$ of each key $k \in \mathcal{K}$,

$$H_0[k] = -\sum_{k \in K} P[k] \log_2 P[k] \tag{1}$$

Maximum entropy is achieved when each key has an equal probability of occurrence, i.e., for uniform random input, $H_0 = \log_2 |\mathcal{K}|$. Comparatively, the entropy of written English

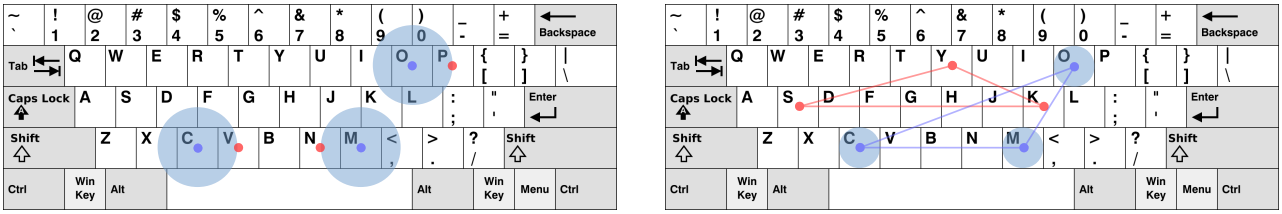

Fig. 4. Isomorphic 3-letter words in 1st order (left) and 2nd order (right) spatial side channels. Shaded blue circles denote measurement error.

is about 2 bits per character for 5-letter words and 1 bit per character for sequences beyond 100 characters [38].

After an attacker has observed some side channel measurement y, the probability of each key having occurred may change. This could be due to, e.g., the localization of an acoustic emanation to either the left or right side of the keyboard. The relative entropy of a keystroke, given a measurement y from a side channel, is

$$H_1\left[k|y=y_0\right] = -\sum_{k\in K} P\left[k|y_0\right]\log_2 P\left[k|y_0\right] \tag{2}$$

where $P\left[k|y_0\right] = \frac{P[y_0|k]P[k]}{\sum_{k\in K} P[y_0|k]P[k]}$ according to Bayes' Theorem. The total relative entropy is given by

$$H_1\left[k|y\right] = \int P\left[y_0\right] H_1\left[k|y=y_0\right] dy_0 \tag{3}$$

where $P\left[y_0\right] = \sum_{k\in K} P\left[y_0|k\right] P\left[k\right]$. The side channel information gain, or *mutual information*, is the difference between the intrinsic entropy and relative entropy,

$$I\left[k;y\right] = H_0\left[k\right] - H_1\left[k|y\right] \tag{4}$$

which specifies how many bits of entropy are leftover after learning a measurement from the side channel. One may also calculate the relative information gain,

$$I_R\left[k;y\right] = I\left[k;y\right]/H_0 \tag{5}$$

which is the ratio of information gained to the intrinsic information: $I_R = 0$ indicates no change from the intrinsic entropy and $I_R = 1$ indicates that a single key has been positively identified, analogous to perfect classification accuracy.

B. Spatial Side Channels

A spatial side channel reveals the physical locations of keys on the keyboard through spatial measurements. Attacks of this kind may be performed through acoustic localization [39], video of the keyboard [40], or WiFi signal distortion induced by hand motion [5], to name a few. While most spatial side channels require an external sensor to obtain spatial measurements, physical key locations can also be sensed through a side channel on the host computer, such as through cache patterns correlated to specific keyboard keys [7].

There exist two types of spatial side channels: 1st order spatial side channels are those that indicate physical key locations, for example by localizing the source of acoustic emanations [39]; 2nd order spatial side channels provide only the distances between physical keys, for example by measuring the acoustic similarity between two different key presses [37].

1) 1st Order Spatial: In a 1st order spatial side channel, sensor measurements reveal physical key locations. However, these measurements may not be exact. That is, the key locations might be known only to within some error. This error could be due either to noise or the resolution of the sensor. For example, attempting to localize the sound of a key press with a microphone that has a 16 kHz sampling rate is accurate only to within 2.1 cm since sound travels at 343 m/s. The error could instead reflect a logical grouping of keys, such as which scan column a key resides within [6]; in this case, each measurement corresponds the set of keys along the same scan column, and these keys may or may not be in spatial proximity. Despite this, substantial information gains can be achieved even with considerable measurement error as demonstrated in this section.

As an example, Figure 4 (left) shows three spatial measurements with ±1 u error observed when the user types the word "com". Let k_1, k_2, k_3 be the sequence of unknown keys to an adversary. Assuming uniform error in every direction, each measurement covers an area with radius 1 u, limiting the number of possibilities for each k_i. That is, $k_1 \in$ {X,C,V}, $k_2 \in$ {I,O,P}, and $k_3 \in$ {N,M}. For randomly-typed letter-only input, such as a password, this limits the number of possible sequences from $26 \times 26 \times 26 = 17576$ ($H_0 = 14.10$ bits) to $3 \times 3 \times 2 = 18$ ($H_1 = 4.17$ bits), an information gain of 9.93 bits (0.70 relative information gain).

If we assume the user is typing an English word, the number of possible sequences is limited even further. Only English words with letter sequences that fall within the measurement error need to be considered. One possibility, shown by the red dots in Figure 4 (left), is the word "vpn" since the V, P, and N keys are each within 1 u proximity to the C, O, and M keys. Generally, the spatial constraints given by this particular measurement can be captured by the regular expression (regex): "^[xcv][iop][nm]$". Matching this regex to a dictionary of the 10k most common English words gives 4 possible results: "com", "con", "von", and "vpn", out of 672 3-letter words, a reduction of $9.39 - 2.00 = 7.39$ bits (0.79 relative information gain, assuming each word occurs with probability $P\left[w\right] = \frac{1}{672}$).

To get a sense of how spatial measurement error affects information gain, the above procedure is used to calculate the relative information gain for each word in a dictionary comprised of the 10k most common words since year 2000 from the Google Web Trillion Word Corpus [41]. Figure 5 (top left) shows the relative information gain as word length

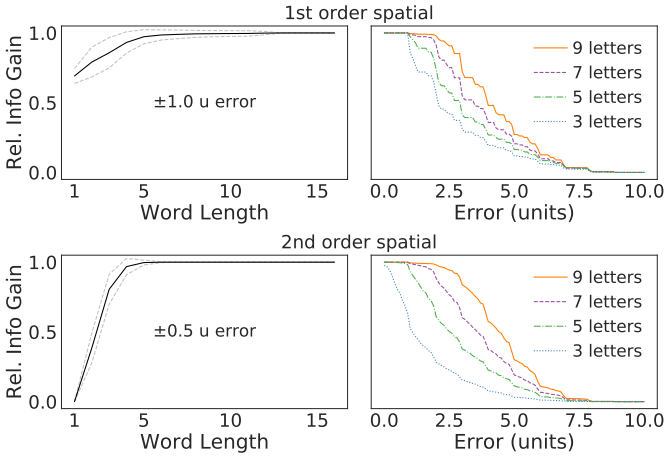

Fig. 5. Spatial info gain vs word length (left) and measurement error (right).

increases given ±1 u measurement error. At 6 letters, words are almost determined with certainty, reflected by the near perfect relative information gain.

Figure 5 (top right) shows the relative information gain of different word lengths for increasing measurement error. At ±4 u, gains begin to drop below 0.50, and beyond ±7 u error, almost no information is gained. Note that 0 information gain is achieved for ±10 u since this radius spans the length of letter keys on the keyboard, and error less than ±1 u can achieve perfect accuracy since no error radius overlaps any other key.

2) 2nd Order Spatial: A spatial side channel is 2nd order if it provides the *distances* between key locations as opposed to the key locations themselves. This may occur if an attacker observes a function that measures the distance or similarity between pairs of physical keys. For example, consider the acoustic emanations captured by a single microphone. Although localization is not possible, different keys have been shown to retain different acoustic signatures [39]. Comparing the acoustic waveform of two different key presses enables an adversary to differentiate between pairs of keys, despite not knowing the key identities. Further, keys that are within spatial proximity typically produce similar sounds which can actually reveal the physical inter-key distances [37]. In this way, a context-free attack can be performed, omitting the need for a pre-trained classifier.

Since a 2nd order spatial side channel uses the distances between keys, it is the key pairs and not individual keys that are recognized. As shown in Figure 4 (right), consider typing the word "com" with distances 5.6 u, 2.4 u, and 4.0 u between keys C-O, O-M, and C-M, respectively. With a measurement error of ±0.5 u, the word "sky" is isomorphic to "com" since it has distances 6.0 u, 2.5 u, and 3.9 u between keys S-K, K-Y, and S-Y, respectively. Generally, for a sequence of n keystrokes, there are $\frac{n(n-1)}{2}$ unique distances in the distance matrix formed by the key pairs.

Using the same dictionary in the previous section, the relative information gain for each word is calculated. First, the inter-key distance matrix for each word is computed. This distance matrix is compared to the distance matrix of every other word of same length in the dictionary, matching words that are within the measurement error. Note that unlike a 1st order spatial side channel, the 2nd order spatial measurement error is over distances and not locations. Figure 5 (bottom left) shows the relative information gain for a 2nd order spatial side channel with ±0.5 u error. The relative information gain for words of 1 letter is 0 since no distances are observed. Figure 5 (bottom right) shows the relative information gain as measurement error increases.

In the worst case, a distance function can be binary valued, in that it indicates only whether two keys are the same or different. This effect could be achieved by thresholding the distance matrix [42], revealing the unique characters in a word but not the characters themselves, much like a monoalphabetic substitution cipher. In this case, words can be matched using an extended regular expression. For example, the regex for "attack" would be "^(.)((?!\1).)\2\1((?!\1|\2).)((?!\1|\2|\3).)$" which matches the words "effect", "attach", "affair", "attain", and "oppose". Note that this method only provides a substantial reduction in the search space if there are letter repetitions. The number of matches for "social" (or any 6 letter word without character repetitions, which have the same regex pattern) is 784 out of 1543 6-letter words, an information gain of about 1 bit.

C. *Temporal Side Channels*

There are two events associated with every keystroke: press and release, with respective timings given by t^P and t^R. A temporal keylogging side channel uses the sequence of keystroke timings, t^P and/or t^R, to predict which keys were pressed, exploiting the consistent and predictable way in which a victim types. As noted by Salthouse, touch typists exhibit a number of phenomena related to the dependence of keystroke timings on physical key placement [43], [44], such as:

- Keys that are far apart are pressed in quicker succession than keys that are close together.
- Letter pairs that occur frequently in language are typed in quicker succession than infrequent letter pairs.
- Practicing a specific keystroke sequence can significantly reduce inter-key timings.

These phenomena, among others, are the basis for exploiting the relative information between physical key locations and keystroke timings. Analogous to spatial side channels, a temporal side channel can reveal either individual keys, through the key-hold duration, or key pairs, through the time intervals between successive keystrokes.

1) Duration: The duration of a keystroke is the time interval from press to release, with the ith duration given by

$$d_i = t_i^R - t_i^P \tag{6}$$

which can be used to identify individual keys. Typists may generally hold down different keys for different lengths of time, permitting an adversary to infer which key was pressed (or which keys were more likely pressed) based on the observed duration. This phenomenon is shown in Figure 6 (left)

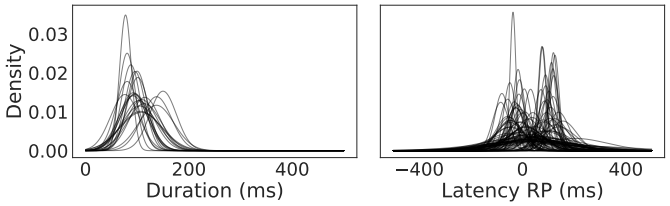

Fig. 6. Duration (left) and latency (right) Gaussian PDFs for a single user.

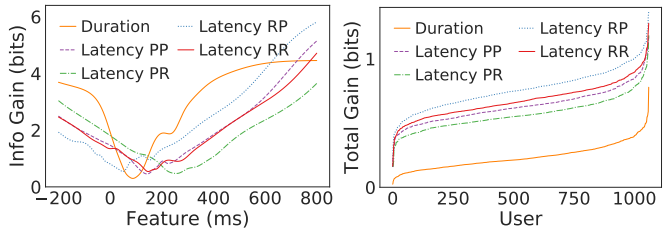

Fig. 7. Info gain for a single user (left) and total info gain per user (right).

with Gaussian PDF estimates for each of 22 unique keys in a sample of English text from a single user in a large keystroke dataset [45] (see Appendix A for dataset details and Appendix B for a summary of public keystroke datasets). The information gain is determined with $P\left[d|k\right]$ given by the Gaussian PDF estimate for each key and $P\left[k\right] = \frac{1}{22}$ to reflect a uniform prior (maximum intrinsic entropy with 22 observed keys). Thus, $H_0 = 4.46$ bits since there are 22 unique keys. In Figure 7 (left), more information is gained for relatively high or low durations since these are rare; information gain is lowest where the durations tend to cluster around 120 ms. The total information gain is 0.36 bits per key.

In many side channels, the time interval between successive keystrokes is of interest. This is especially true for those attacks described in Section IV-D that exploit network traffic timings in which only key press timestamps are available. For these attacks, duration cannot be computed because release times are not available.

2) Latency: The latency between successive keystrokes can be utilized to identify key pairs. There are exactly four different latency features formed by each combination of key press and release from two successive keystrokes, given by

$$\begin{aligned} \text{Press-Press (PP):} \quad & \tau_i^{PP} = t_i^P - t_{i-1}^P \\ \text{Press-Release (PR):} \quad & \tau_i^{PR} = t_i^R - t_{i-1}^P \\ \text{Release-Press (RP):} \quad & \tau_i^{RP} = t_i^P - t_{i-1}^R \\ \text{Release-Release (RR):} \quad & \tau_i^{RR} = t_i^R - t_{i-1}^R \end{aligned} \tag{7}$$

Unlike duration, the latencies correspond to pairs of keys, or bigrams. Despite the intrinsic entropy being much larger (there are $26 \times 26 = 676$ possible bigrams from letter keys alone), information gained from latency can be substantial due to a greater separation between bigram latency distributions.

Figure 6 (right) shows the Gaussian PDF estimates for RP latency from the same user in the previous section. Like duration, PP and PR latencies are nonnegative. However, RP and RR latencies can be negative when the release of the first key overlaps the press or release of the second key, respectively (commonly encountered for modifier keys, such as Shift). The information gain for each latency feature is shown in Figure 7 (left), where intrinsic entropy $H_0 = 6.57$ bits since there are 95 unique bigrams. For this particular user, each latency feature provides about twice as much total information as duration. Again, information gains are greatest for extremely large or small latency values since these occur less frequently.

TABLE I
PEARSON'S CORRELATION BETWEEN INFORMATION GAIN AND TYPING SPEED. BOLD p-VALUES INDICATE SIGNIFICANCE AT A 0.05 THRESHOLD.

	Du	**PP**	**PR**	**RP**	**RR**
Pearson's r	-0.156	-0.011	-0.113	0.211	0.107
p-value	**3.6e-07**	7.3e-01	**2.3e-04**	**3.9e-12**	**5.7e-04**

3) Users and Typing Speed: The previous sections examined information gain for a single representative user. However, since a temporal side channel exploits the predictable way a user types, information gain could vary with different users based on different typing speeds and styles [46]. Using the same procedure to calculate I, Figure 7 (right) shows the per-user information gains for each of the 1060 users in the same dataset [45]. For the majority of users, duration provides between 0.15 and 0.30 bits of information and each latency between 0.5 and 0.9 bits with RP latency usually providing the most. These results suggest that temporal side channel information gain is highly user-dependent.

A question then arises as to what user-dependent factors might determine the amount of information that can be gained through a temporal side channel. We examined the relationship between information gain and typing speed as given by the number of keystrokes per minute (KPM), excluding non-letter keys and latencies over 1 s. We found that the information gained from duration decreases with increasing typing speed, and for latency features, the opposite is generally true. Except for PP latency, the correlations are significant, with Pearson's r and the corresponding p-value for each feature summarized in Table I. Corroborating this relationship, an examination of the data reveals that as typing speed increases, keystroke durations become more uniform and consistent leading to reduced information gain. Conversely, the dependence of inter-keystroke timings on key distance and location becomes more evident with increased typing speed, as previously noted by Salthouse and others [43], [47].

IV. ATTACKS

A keylogging side channel attack can target either the user, the keyboard itself, the host computer, or the network. Table II provides a summary of specific keylogging side channel attacks for each of these targets where each attack is characterized by the following attributes.

Modality is the medium through which the side channel is sensed, such as a physical medium (e.g., sound), a hardware component (e.g., shared memory), or an application protocol.

TABLE II

KEYLOGGING SIDE CHANNEL ATTACKS. RAD=RADIATIVE, CAP=CAPACITIVE; S1=1ST ORDER SPATIAL, S2=2ND ORDER SPATIAL, T=TEMPORAL; ○=SLOW (≤120 KPM), ●=NORMAL (>120 KPM); □=WITHIN-SUBJECT, ■=BETWEEN-SUBJECT, ☒=NONE; -=UNKNOWN/NOT PROVIDED.

	Modality	Attack Vulnerability	Proximity	Channel Type	Typing Speed	Requires Training	Performance		Ref.
							Detection	Identification	
User	EEG	Decode brain electrical activity	Headset	S1	●	□	-	63% key ACC	[48]
	Motion	Hand location over keyboard	Smartwatch	S1/T	●	■	57% TPR	30% word ACC5	[49]
	Motion	Hand movement over keyboard	Smartwatch	S1/S2	●	☒	-	80% PIN ACC	[50]
	Motion	Hand movement+key acoustics	Smartwatch	S1/S2	●	■	-	55% word ACC5	[51]
	Motion	Hand movement+key acoustics	Smartwatch	S1/S2	●	■	-	51% word ACC10	[52]
	Video	Line of sight to keyboard	< 1 m	S1	●	☒	-	46% word ACC	[40]
	WiFi	WiFi CSI distortion patterns	4 m	S1	○	□	98% TPR	96% key ACC	[5]
	WiFi	WiFi multipath localization	5 m	S1	○	□	-	92% key ACC5	[53]
Keyboard	Acoustic	Keyboard acoustics	1 m	S1	○	□	-	79% key ACC	[54]
	Acoustic	Keyboard acoustics	-	S1	●	■	-	64% key ACC	[55]
	Acoustic	Keyboard acoustic differences	-	S2	-	☒	-	73% word ACC50	[37]
	Acoustic	Bootstrapped keyboard acoustics	-	S1/S2	●	☒	-	90% word ACC	[56]
	Acoustic	TDoA localization (3 mics)	< 1 m	S1	●	☒	-	72% key ACC	[39]
	Acoustic	TDoA localization (2 mics)	< 1 m	S1/S2	○	☒	-	94% key ACC	[57]
	Acoustic	Keyboard acoustics through VoIP	Remote	S1	●	□	-	83% key ACC	[58]
	Acoustic	Keyboard acoustics through VoIP	Remote	S1	○	□	-	74% key ACC	[9]
	EM Cap.	PS/2 wire crosstalk	15 m	S1	-	☒	-	-	[42]
	EM Rad.	PS/2 signal radiation	15 m	S1	-	☒	-	95% key ACC	[6]
	EM Rad.	Matrix scan delay position	5 m	S1	-	☒	-	95% key ACC5	[6]
	Seismic	Vibration sensed by acoustic laser	30 m	S2	-	☒	-	-	[42]
	Seismic	Vibration sensed by smartphone	50 mm	S2	○	□	-	43% word ACC10	[59]
Host	`procfs`	`procfs stats` (ESP and EIP)	Kernel	T	-	□	100% TPR	40% word ACC10	[60]
	`procfs`	`procfs schedstat`	Kernel	T	-	☒	-	-	[61]
	CPU	Shared event loop time differences	Browser	T	●	☒	98% TPR	-	[62]
	CPU	Instruction throughput differences	Browser	T	●	□	-	79% word ACC	[29]
	CPU	`rdtsc` differences	Core	T	●	☒	100% TPR	-	[10]
	CPU	X11 event duration	Core	S2	-	☒	-	2.5bits/key	[63]
	CPU	Keyboard interrupt duration	Core	T	-	☒	-	1bit/key	[63]
	Memory	Prime+Trigger+Probe	Cache	T	-	☒	95% TPR	-	[64]
	Memory	Cache-hit ratio pattern	Cache	S1	-	☒	-	3.3bits/key	[7]
	Memory	`clflush` latency	Cache	T	●	☒	92% TPR	-	[65]
	Memory	DRAM row buffer latency	DRAM	T	-	☒	100% TPR	-	[66]
	Memory	Multi-Prime+Probe	Cache	T	●	☒	92% TPR	-	[10]
	EM Cap.	USB hub crosstalk	USB hub	S1	●	☒	-	97% key ACC	[67]
Net.	HTTP	HTTP response size	Remote	S1	●	☒	-	3.6bits/key	[68]
	SSH	Packet timing (interactive shell)	Remote	T	●	■	-	1bit/key	[8]
	VoIP	RTP event packet timing	Remote	T	●	■	-	1.7bits/key	[69]

Proximity indicates either the physical distance of the sensor to the victim, or the shared resources required by the attack, such as being resident on the same CPU core. Network-based attacks are considered *remote* since they don't require any any physical sensor or code execution on the victim's host.

Channel Type can be either 1st order spatial (S1), 2nd order spatial (S2), temporal (T), or combination thereof, as described in Section III. The type of side channel is determined by both *what information is sensed* and *how that information is used*. To demonstrate this, consider how the acoustics from a single microphone can be used in three different ways:

- S1: the attacker identifies individual keys or groups of keys using a classifier that was trained on a separate labeled dataset (supervised approach).
- S2: the attacker compares the acoustics between pairs of keys to form a set of constraints and then performs a dictionary lookup (unsupervised approach).
- T: the attacker extracts the key press and release timings from the acoustic signal and then identifies keys and key pairs based on the time intervals.

Typing Speed refers to the typing speed of the victim during an attack. Some attacks have been demonstrated as a proof of concept by making restrictive assumptions on the typing speed of the victim, such as pressing only a single key at a time in a slow manner, i.e., the "straw man" approach [55], while other attacks operate at a normal typing speed. The exact conditions of each attack vary in many ways besides typing speed, but this metric provides a general idea as to the use case considered.

Requires Training indicates whether the attack requires a separate labeled dataset for supervised training. Some attacks require a classifier trained on keystrokes collected from the same victim or keyboard, e.g., obtained through social engineering, as opposed to a dataset which has been crowdsourced. These scenarios are referred to as *within-subject* and *between-subject*, respectively, to reflect whether the victim or keyboard must be present in the training dataset.

Performance is reported using the metrics described in Section III-A: the TPR for keystroke detection (most attacks either do not report a TNR or assume this to be 100%) and information gain for key identification. Instead of information

gain, some attacks report the rank-n identification accuracy (ACCn), which is the probability of correctly identifying the correct key or word among the top-n choices as ordered by the classifier. Note that key, as opposed to word, identification operates at the character level and implies the method can be applied to arbitrary (e.g., password) input, while word identification assumes the user types a dictionary word.

A. *Attack the User*

A side channel attack targeting the user relies on sensing the user's physical state during typing. Four such modalities have been exploited, including electroencephalography (EEG), the motion of the wrist as sensed through a smartwatch, video with keyboard line-of-sight, and WiFi signal distortion.

1) Wearable Devices: Electrical potential differences emanating from the superficial layers of the brain are detected on the scalp by an EEG cap, which are becoming more prevalent as they decrease in cost. EEG has long been thought to reflect a user's cognitive state [70], such as by detecting the characteristic P300 response to known stimuli [71], and just recently has been considered as a modality for eavesdropping keyboard input [48]. Other wearable devices that exploit the motion of a victim's hands while typing, such as smartwatches and fitness trackers, present similar privacy concerns [72]. Such devices are equipped with most of the same sensors found in smartphones, and despite the sampling rates being slightly lower (on the order of 50 Hz, compared to 100 Hz on smartphones), the accelerometer enables a fairly precise estimate of relative hand distance traveled with less than 1 cm error [50]. However, there are some unique challenges, notably that when worn on the left hand a smartwatch is essentially blind to the movements of the right hand.

2) Video: The ability to simply view the keyboard while a victim types might seem like the ideal method to obtain perfect key identification, but in practice this is more difficult to achieve. It takes a human analyst anywhere from 1–2 hours to identify the keystrokes in a 3 minute video, and only with about 90% key identification accuracy [40]. Such difficulties arise from occlusions by the hand, simultaneous movement of alternate hand and fingers, and the typing speed of the victim. This process can be automated using traditional computer vision processing techniques (segmentation and motion detection), assuming the attacker has access to a compromised webcam pointed at the keyboard in close proximity ($<$1 m).

3) WiFi: WiFi signals have traditionally been used for coarse grained gesture recognition [73] and only recently considered as a keylogging side channel whereby the user's finger and hand movements are localized with high enough resolution. In a "straw man" approach, finger location can be detected through the induced changes in signal delay at two receiving antennas [53] or by fluctuations in the instantaneous channel state information (CSI) [5], [74], which describes the signal propagation characteristics. The former method requires a software-defined radio (SDR) and multiple antennas at fixed distance, in the same spirit as sound source localization, and the latter approach uses commercially-available hardware.

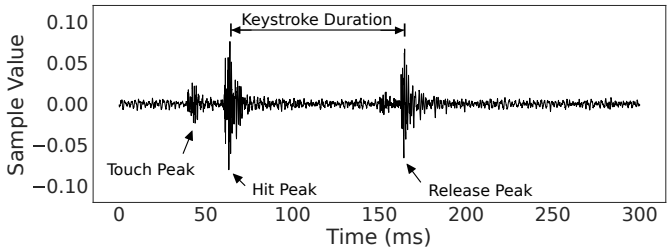

Fig. 8. Keystroke acoustics. Peaks occur at the touch, press, and release.

B. *Attack the Keyboard*

The keyboard emits a variety of unintended signals, including acoustic, seismic, and electromagnetic emanations. As such, there are a range of attacks that exploit side channels emanating from the keyboard. For these attacks, a physical sensor external to the host computer is almost always utilized.

1) Acoustic: Acoustic side channels represent a majority of keylogging side channel attacks. Under normal typing conditions, most keyboards emit a characteristic acoustic signal upon each keystroke, which can be captured up to several meters away with a omnidirectional microphone and up to 15 meters with a directional microphone [54]. An example of this waveform, shown in Figure 8, has three distinct parts: the touch, press, and release. The touch peak (42 ms) occurs when the user's finger makes contact with the surface of the key, but before the key is pressed; the press peak (64 ms) occurs when the key is pressed and makes contact with the underlying plate in the keyboard; and the release peak (165 ms) occurs when the key settles back into its upright resting position.

Due to the characteristic peaks emitted, keystroke detection rates using acoustic methods are generally high and assumed to be 100% TPR in many works. Without too much background noise, simple detection methods, such as the energy in a sliding window exceeding a given threshold, are effective [37], [55]–[57]. Despite the ability to extract high-resolution timings from an acoustic signal, most acoustic side channel attacks only utilize spatial information. Keys are identified in primarily two ways: through sound source localization and by comparing the acoustics produced by different keys.

With at least three microphones, acoustic emanations can be localized by *multilateration*, leveraging the time difference of arrival (TDoA) of the key acoustics to each receiver [39]. The key identification accuracy of such an approach depends on the sampling rate of the microphones which introduce an *inevitable error*, the degree to which noise-free localization can be performed. A 44.1 kHz microphone, common on most smartphones, has about 0.77 cm localization error [39], calculated by $(343\text{m/s}) \,/\, (f_s\text{Hz})$ where f_s is the sampling frequency and 343 m/s is the speed of sound. Partial localization can be performed with two microphones (conveniently located on most smartphones), albeit with much higher error, narrowing down the possibilities to a subset of keys that fit within a hyperbolic window over the keyboard [57].

Different keys on a keyboard tend to emit different acoustic

signatures and can be identified in this way using only a single microphone [54]. Having previously recorded the acoustics of a specific keyboard, i.e., supervised approach, individual keys can generally be identified by comparing the unknown waveforms to the known waveforms collected during training [54]. The acoustic signature of each key has been demonstrated to be consistent enough to log keystrokes not just across users, but across devices of the same model [58] and over VoIP services [9], [58]. Alternatively, a set of constraints on the keystroke sequence can be specified by the acoustic similarity of key pairs. This implies a 2nd order spatial side channel in which the constraints are used to perform a dictionary lookup.

2) Seismic: Similar to acoustic emanations, the motion of the keyboard induced from typing causes minute vibrations in the underlying surface which may be carried over short distances. The detection and recognition of keystrokes based on vibrations can be performed either with a laser microphone, up to 30 m away [42], or through a compromised mobile device in close proximity to the keyboard. Only recently has the sampling rate of such mobile sensors increased to the point at which keystrokes can be reliably identified, with most modern devices capable of reaching 100 Hz [59]. At 50 Hz, previous generation devices were unable to detect, let alone identify, keystrokes from seismic activity.

3) Electromagnetic: There are primarily two sources of EM emanations in a standard keyboard: the keyboard scanning matrix (described in Section II-C), and the communication protocol (PS/2 and USB, described in Section II-D).

As the microcontroller pulses each column in the matrix, an EM spike is emitted. This occurs continuously, and when no keys are pressed, the time interval between EM spikes is about the same. However, when a keystroke is detected, the microcontroller enters a subroutine to encode and transmit the keystroke event. Thus, a short delay is produced, and based on the position of this delay, an attacker can determine which column the keystroke occurred within [6].

Attacks on the communication protocol are particularly effective, often narrowing down to a single key or very small subset of keys by decoding the unencrypted signal, which is carried either through the air or a conductive ground. In PS/2, a strong EM spike is emitted upon each falling edge of the data signal, enabling keys to be identified by their "falling edge" scancode patterns. For example, the bit pattern for E (scancode 0x24) is: 00010010011 which has falling edge pattern: ↓↑↑↑↓↑↑↓↑↑↑, where ↓ denotes the presence of a falling edge in the data signal and ↑ an absence. The G key (scancode 0x34) has the same falling edge pattern since its bit pattern is 00010110001. This spatial attack narrows down to a subset 2.1 alphanumeric keys on average [6].

C. Attack the Host

With the ability to execute code on the victim's machine, there exist a variety of exploitable side channels at the attackers disposal, most of which are in the realm of microarchitectural attacks. The three main modalities are: process footprint as reported by the kernel, CPU load, and memory access patterns. For a full survey of microarchitectural attacks, which include the latter two modalities, see [75].

1) Process Footprint: The kernel itself can leak a considerable amount of information through process state and usage statistics. On Linux, the virtual file system `procfs` is the source of several side channels. Monitoring the values of the extended stack pointer (ESP) and extended instruction pointer (EIP) as reported by `/proc/[pid]/stat` reveal system call patterns to known locations in memory. The way an application responds to keyboard interrupts induces a specific pattern of system calls such that keystroke events can be detected. This attack works on multi-user systems and despite the relatively slow update rate of `procfs` [60]. The scheduling statistics as reported by `/proc/[pid]/schedstat` can also indicate when a process responds to keyboard events since most text-editing applications remain idle until user input is available [61]. An attacker simply counts the number of time slices that a process has been allocated, which, for textual applications, will generally only increase upon each keystroke.

2) CPU Load: The general approach of CPU load-based attacks is to detect spikes in CPU activity induced by IO interrupts. On an otherwise idle system, a key press or release event (or any IO event for that matter), causes a spike in CPU load. The first kind of such an attack measured the duration of each interrupt with a high-resolution timer and found that interrupts for key press and release events had distinct durations when compared to, e.g., scheduling interrupts [63]. Cache usage can also be used to detect spikes in CPU load induced by keystroke events [64], and in some cases the memory footprint can actually reveal individual keys (described in the next section).

CPU load attacks can be performed in sandboxed environments despite being unable to execute native instructions and without access to pointers and high-resolution timers. As an alternative to an explicit high-resolution timer, an attacker can increment a variable inside a (possibly segmented) loop and then measure the number of variable increments at coarser-grained intervals. This effectively measures the number of instructions executed within a given time interval, which offers relatively high-resolution timestamps of keyboard events [29]. Measuring the time between event processing in a shared event loop additionally provides some insight to spikes in web browser activity which may result from keystroke input [62].

Most CPU load attacks are categorized as temporal side channels since they merely indicate the presence of a keystroke and provide little or no spatial information. An exception to this is an attack in which the key map is modified by the attacker such that a particular key takes longer to process than any other [76]. The presence of this key can then be detected in a password by monitoring the target application execution time (e.g., xlock) when a keystroke event is received: a long execution time indicates the modified key was typed by the victim and a short execution time implies its absence. This process is repeated for each key in the key map or until all the keys in a password have been identified. However, this intentionally causes the wrong character to be typed, possibly alerting to the victim to the presence of the attack.

3) Memory Footprint: Cache attacks leverage the nearly-ubiquitous design of shared cache and memory to detect the use of specific memory addresses by a target application. An attacker can discern which locations are or are not accessed by a target application by measuring the time it takes to access a specific address mapped to the same cache set. There are several variations of this general approach in which the cache set is primed by the attacker before invoking (or waiting for) the target application and then later probed to measure latency [77]. Since cache attacks can detect which memory locations are accessed, as opposed to just CPU load, they can be used as a spatial side channel and potentially determine which keys were pressed. Such is the approach of the cache-hit ratio template that characterizes which addresses are frequently accessed by a target library or binary for each key [7]. Even the row buffer in DRAM, which acts as a kind of cache, is susceptible to this type of attack [66].

4) USB Crosstalk: Unlike radiative coupling, capacitive coupling exploits the undesirable transfer of energy between electrical components in close proximity. The keyboard state, transmitted in 8-byte frames using NRZI encoding (see Section II-D), is clearly visible to neighboring USB devices through both the data and power lines [67]. Using relatively simple signal processing techniques, a malicious USB device is capable of eavesdropping on upstream USB 1.x and 2.0 traffic from neighboring devices connected to the same hub.

D. *Attack the Network*

The client-server programming model has come to dominate web-based interactive applications. Although many web applications utilize encrypted communications, most do not take any measure to obfuscate the communication patterns that manifest. This is especially problematic for applications that wait for user input, as each network packet itself may correspond to a keystroke revealing both the key press time of the victim and the payload size of the server response.

1) Payload: In some cases, keystrokes can be identified by the size of a response from the server in a web-based application [68]. This affects applications that implement real-time autocomplete suggestions, whereby each suggestion has a unique size. As the server responds to each query consisting of only a single keystroke, it provides a uniquely-sized list of responses which yield considerable information gains.

2) Timing: Network timing attacks implicate a broad range of real-time client-server applications, including those not susceptible to any form of payload analysis. As the victim types in an otherwise idle application, the client emits bursts of network traffic which, to any listening adversary, can reveal the key press timings of the victim. The key press latencies can be used to either reconstruct the victim's input from a dictionary, or to guide a search in password cracking [8], [69]. There is, however, some debate as to whether this temporal keylogging side channel is damaging in practice. The attacker must be able to distinguish between network traffic generated by key presses and other unrelated traffic. With background traffic, such a task may become impractical [78], [79].

E. *Exfiltration*

Attackers face the additional problem of data exfiltration which, often neglected, can be more challenging than keylogging itself. Methods of exfiltration differ primarily based on whether the attack was mounted on a device controlled by the user (e.g., smartwatch or host computer) or the attacker (e.g., a microphone or antenna). The former scenario is especially challenging since exfiltration must be performed without alerting the user to the attacker's presence. So as to minimize the risk of being detected, a covert channel may be established to retrieve the logged keystrokes. Of particular note is JitterBugs, a class of covert channels designed specifically for keystroke exfiltration [80]. JitterBugs establishes a covert channel by modulating the keystroke timings themselves, which can later be remotely detected over a network during an interactive application. In a base-2 encoding scheme, each key press transmits a single bit by aligning its time interval to a multiple of some modulus m: time intervals close to $0 \mod m$ are bit 0 and close to $\frac{m}{2} \mod m$ are bit 1. This could be performed either in software at the driver level, or in hardware as a buffering device placed between the keyboard and the host.

V. Defenses

Completely eliminating some keylogging side channels has proved to be an elusive goal [91], necessarily a consequence of the difficulty to analytically describe any physical system [92]. While protocols and design specifications may be built deductively from a set of axioms, the physical components that embody such a system are subject to noise, interference, and unforeseen side effects. This leaves the possibility, however small, that the system may not behave as intended. In this sense, a keylogging side channel defense *mitigates* the possibility of an attack under a specific set of assumptions, and not in general. There are primarily three different approaches.

Impediment: A defense that restricts access to a sensor or eliminates unwanted emanations can be said to *impede* an adversary from observing the compromising signal. This can be implemented digitally, such as through permissions-based access control, or physically, such as by shielding or suppressing the compromising emanations.

Obfuscation: Decreasing the signal-to-noise ratio can *obfuscate* the side channel, rendering a particular attack ineffective. With this approach, an adversary may still observe the signal, but the information content is too low for keystroke detection and/or key identification. Obfuscation can be achieved by increasing background noise or decreasing the sensor resolution.

Concealment: The presence of superfluous information may *conceal* the side channel from an adversary. With this approach, the original signal is left intact but becomes indistinguishable from irrelevant overlapping signals aimed to mask the true keyboard events, consequently making keystroke detection much more difficult. Since keystroke detection is a prerequisite for key identification, a defense that mitigates the former is also generally effective against the latter. Like the other approaches, concealment could be implemented digitally,

TABLE III
KEYLOGGING SIDE CHANNEL DEFENSES. DET=KEYSTROKE DETECTION, ID=KEY IDENTIFICATION.

	Modality	Defense	Method	Target	Channels Protected			Noticeable to User?	Ref.
					S1	S2	T		
User	EEG	Induce covert responses to irrelevant stimuli	Obfuscate	ID	✓	✓		✓	[71], [81]
	EEG	Filter keystroke-identifying features	Impede	DET	✓	✓	✓		[71], [82]
	Motion	Limit sensor permissions during typing	Impede	DET	✓	✓	✓		[51]
Keyboard	Acoustic	Reduce keyboard acoustic emanations	Impede	DET	✓	✓	✓	✓	[54]
	Acoustic*	Keys produce homomorphic sounds	Obfuscate	ID	✓	✓			[54]
	Acoustic*	Emit synthetic keyboard sounds	Conceal	DET	✓	✓	✓	✓	[9], [83]
	EM Rad./Cap.	Filter/shield EM emanations	Impede	DET	✓	✓	✓		[3], [84]
	EM Rad.	Randomly delay matrix scan routine	Obfuscate	DET	✓	✓	✓		[85]
	EM Rad.	Randomize matrix scan pattern	Obfuscate	ID	✓	✓			[86]
Host	CPU/Memory	Generate spurious key press/release events	Conceal	DET	✓	✓	✓		[10]
	CPU/Memory	Decrease timer resolution	Obfuscate	DET			✓		[87], [88]
Net.	HTTP	Obfuscate packet size through padding	Obfuscate	ID	✓	✓			[68]
	SSH/VoIP	Randomly delay key press/release events	Obfuscate	ID			✓	Maybe	[89], [90]

*Not including acoustic TDoA localization attacks.

by generating spurious keyboard events, or physically, such as emitting synthetic keyboard sounds.

In addition to methodology, each defense in Table III is characterized by the types of side channels they *protect* against (spatial and/or temporal), whether they *target* keystroke detection and/or key identification, and whether they produce any *noticeable* side effects to the user, such as increased noise or changes in application behavior.

A. Defend the User

Dynamic access control to emanating sensors on wearable devices represents an effective defense that is transparent to the user. Such a scheme could either limit sensor permissions while the user is typing [51] or filter the features that permit keystroke detection [71]. This form of impediment restricts a malicious application from detecting the user's keystrokes without sacrificing usability, although it requires the device to reliably detect when the user is typing.

It might also be possible for a user to "trick" the device by modifying their own behavior while typing as a form of obfuscation. Wearing an EEG cap, this could be accomplished by inducing covert responses to irrelevant stimuli, such as by thinking of a different key than the key that is physically pressed [71]. Some work in this area indicates that systems using the P300 event related potential (ERP) for stimulus detection can be defeated in this way [81], however EEG key identification likely leverages electromyogram (muscle) artifacts induced by hand and eye movement [48].

B. Defend the Keyboard

Several defenses were adopted early on in response to the TEMPEST threat, such as signal filtering and protective shielding, both aimed to impede EM emanations [3]. Official EM radiation policies mandated a 200 ft perimeter to be secured around vulnerable devices, a somewhat arbitrary choice determined to be the largest manageable radius. Interestingly, they also suggested that operating at least 10 devices in parallel could instead be used as a form of concealment [2], in the same spirit as some host-based defenses recently developed, e.g, KeyDrown [10]. For commodity keyboards, filtering the high frequency emissions of matrix scanning may suppress the EM spike that enables column identification [84]. Likewise, randomizing the scan pattern [86] or inserting random delays into the digitization routine [85] would mitigate column identification and keystroke detection, respectively.

The three different approaches to mitigation (impediment, obfuscation, and concealment) are well captured by the various acoustic defenses. As a form of impediment, a completely "quiet" keyboard, one that emits no acoustic emanations, would prevent all kinds of acoustic attacks despite having a noticeable effect of lacking auditory feedback [54]. Instead, a keyboard that obfuscates key acoustics by producing a homogeneous sound for each key would make key identification difficult, although such a device may be difficult to fabricate [54]. This approach is also not effective against multi-mic TDoA localization attacks which do not make use of individual key acoustics. Finally, concealment could be achieved by emitting spurious keystroke sounds in proximity to the user during typing [9], [83], however also potentially failing against TDoA localization attacks if signal separation can be performed based on source location.

C. Defend the Host

Given the number and complexity of shared resources on modern computing devices (CPU, memory, etc.), host-based attacks, especially those that leverage microarchitectural side effects, are remarkably pervasive [93]. Decreasing timer resolution can prevent some forms of keystroke detection, such as those that detect spikes in CPU load [87], however there remain numerous other side channels that can achieve the same effect without explicit high-resolution timers [91].

A concealment-type defense may be more appropriate for such attacks targeting the host. By generating many spurious keystrokes, which appear indistinguishable from the true keystrokes, a user can evade keystroke detection by a malicious application. This is the approach of KeyDrown, a three-layer model that aims to protect against both CPU-load and shared memory microarchitectural attacks targeting the kernel,

shared library, and application layers [10]. With relatively little overhead, the artificial input events follow the same execution path as the true keystrokes, degrading the practical detection TPR to the point of random guessing.

D. Defend the Network

Padding represents a broad class of obfuscation-type defenses against side channels that leverage network packet size. However, where to pad (e.g., HTTP header vs body) and the specific strategy that should be applied (e.g., padding to a quantized length vs padding by random amounts) depends on the particular application [68]. Given the wide range of semantics in web application traffic, compared to, e.g., SSH, a general solution seems nontrivial.

Most network timing attacks, on the other hand, can be prevented to a degree by introducing a small random delay to the keyboard events by temporarily buffering the event on the host or the keyboard itself [89], [90]. This random delay obfuscates the actual time intervals between successive keystrokes, effectively reducing the mutual information between the keystroke latencies and bigrams. The caveat is that it also introduces an additional latency between the user and the application which, if too large, may be noticeable to some users. Longer delays enable greater obfuscation ability at the expense of a reduction in perceived application responsiveness.

VI. Discussion

The keylogging side channel attacks in Table II summarize nearly two decades of research, which has it roots toward the end of WWII [2] and seems to have advanced considerably behind closed doors during the Cold War era as evidenced by revelations such as the Selectric Bug [1]. Comparatively, there is a much smaller body of research directed towards keylogging side channel defenses (Table III), none of which have been widely adopted. This may change in the near future, as some low-cost (in terms of usability, performance overhead, and cost of deployment) countermeasures against host-based and network-based attacks have recently emerged [10], [89].

It is worth noting, keylogging attacks that exploit device behavior, such as EM emanations and cache usage, generally achieve higher performance than those that exploit human behavior, such as smartwatch motion and packet timing. Arguably, this gap is due to differences in regularity between user and device behavior. An attack that exploits microarchitectural side effects is expected to work across all devices of the same make and model, dependent on the highly-consistent behavior across devices. In contrast, an attack that exploits human behavior must adapt to changes behavior over time (non-stationarity) and between users. These observations allude to a fundamental relationship between behavior homogeneity and attack severity with regard to side channels, whereby users and devices who behave contrary to the norm are rewarded by being less susceptible to attack. On the other hand, doing so may actually compromise anonymity, i.e., enable behavior-based identification, an issue not considered in this work.

Temporal keylogging side channels actually exploit a well-established phenomenon in transcription typing, that is, *different users can be expected to operate a keyboard within similar time constraints*, enabling an adversary to make general inferences about user actions based on temporal behavior. For the touch typist, shorter time intervals usually correspond to keys that are far apart compared to longer time intervals for keys that are close together. This is a result of having to reuse the same finger or hand for neighboring keys while distant keys are pressed in quicker succession through parallel processing by alternate fingers [43]. Consistent with this phenomenon, in Section III-C3, we found typing speed to be somewhat indicative of how susceptible a user is to a temporal attack whereby the faster touch typists were more vulnerable. This dichotomy in performance is reminiscent of the *biometric menagerie*, which specifies that biometric identification systems work well for some users (sheep) but are problematic for others (goats) [94]. Given that some users are more susceptible to temporal attacks than others, a better understanding of what other user-specific factors influence temporal information gain could shed some light on new effective countermeasures.

Finally, consistency in HCI behavior across a population is not restricted to typing, and the presence of temporal keylogging side channels reflects a much broader problem potentially faced by interactive client-server applications. Similar phenomena exist with other modalities, such as Fitts' Law in navigating the mouse pointer on a computer screen [95] and more recently Finger-Fitts' (FFitts) Law for touch screen behavior [96]. This raises the question as to what other HCI modalities are subject to such attacks whereby temporal patterns can reveal user actions, and what defenses must *necessarily* be deployed to mitigate this class of *human-based timing side channels*. Given the proliferation of real-time client-server applications, where the real-time constraint dictates that human input events must propagate to the network layer, this issue warrants further investigation.

VII. Conclusion

The keyboard is ubiquitous in human-computer interaction. Even with alternatives, such as voice and eye movement, the keyboard remains an integral device for textual input. This is likely due to familiarity, speed, and the relative accuracy with which the keyboard can be operated. Keylogging side channels will likely remain just as ubiquitous due to the increasing complexity and sensing capabilities of computing devices.

Physiological signals [48] and microarchitectural side effects have proven to be especially pervasive [93], with recent attacks highlighting the copious number of ways to eavesdrop keyboard input. User actions captured through the keyboard and other peripheral devices permeate network communication channels and can have unforeseen side effects on the device. The difficulty in mitigating these types of attacks highlights the importance of empirical studies to evaluate device security [92] and calls for deeper understanding of the security and privacy implications in human-computer interaction.

ACKNOWLEDGEMENT

I thank my colleagues at ARL for suggestions on an early draft and the anonymous referees for their helpful comments.

REFERENCES

[1] S. A. Maneki, *Learning from the Enemy: The GUNMAN Project*. Center for Cryptologic History, National Security Agency, 2012.

[2] J. Friedman, "Tempest: A signal problem," *NSA Cryptologic Spectrum*, vol. 35, p. 76, 1972.

[3] D. G. Boak, "A history of us communications security," *NSA*, 1973.

[4] "IBM Selectric Bug." http://web.archive.org/web/20170311001300/http://www.cryptomuseum.com/covert/bugs/selectric. Accessed: 2017-03-31.

[5] K. Ali, A. X. Liu, W. Wang, and M. Shahzad, "Keystroke recognition using wifi signals," in *Proc. 21st Annual Intl. Conf. on Mobile Computing and Networking (MobiCom)*, pp. 90–102, ACM, 2015.

[6] M. Vuagnoux and S. Pasini, "Compromising electromagnetic emanations of wired and wireless keyboards.," in *Proc. 18th Usenix Security Symp.*, pp. 1–16, USENIX Association, 2009.

[7] D. Gruss, R. Spreitzer, and S. Mangard, "Cache template attacks: Automating attacks on inclusive last-level caches.," in *Proc. 2015 Usenix Security Symp.*, vol. 15, pp. 897–912, 2015.

[8] D. X. Song, D. Wagner, and X. Tian, "Timing analysis of keystrokes and timing attacks on ssh.," in *Proc. Usenix Security Symp.*, vol. 2001, 2001.

[9] A. Anand and N. Saxena, "Keyboard emanations in remote voice calls: Password leakage and noise(less) masking defenses," in *Proc. 8th ACM Conf. on Data and Application Security and Privacy (CODASPY)*, ACM, 2018.

[10] M. Schwarz, M. Lipp, D. Gruss, S. Weiser, C. Maurice, R. Spreitzer, and S. Mangard, "Keydrown: Eliminating keystroke timing side-channel attacks," in *Proc. Network and Distributed System Security Symp (NDSS)*, 2018.

[11] L. Cai and H. Chen, "Touchlogger: Inferring keystrokes on touch screen from smartphone motion," in *Proc. Usenix Summit on Hot Topics in Security (HotSec)*, vol. 11, pp. 9–9, 2011.

[12] H. C. H. Society, *The Story of the Typewriter*. Herkimer, NY, 1923.

[13] "Office Machines and Supplies - Alphanumeric Machines - Keyboard Arrangement," standard, American National Standards Institute (ANSI), May 2009.

[14] "Ergonomics of human-system interaction – Part 410: Design criteria for physical input devices," standard, Intl. Organization for Standardization (ISO), Geneva, CH, Mar. 2008.

[15] "Keystroke sensing." http://web.archive.org/web/20171030204051/https://deskthority.net/wiki/Keystroke_sensing. Accessed: 2017-10-30.

[16] D. Claudio, "Hall effect keyboard," Oct. 5 1971. US Patent 3,611,358.

[17] G. English, "Computer keyboard with flexible dome switch layer," May 18 1993. US Patent 5,212,356.

[18] M. J. Bufton, R. W. Marklin, M. L. Nagurka, and G. G. Simoneau, "Effect of keyswitch design of desktop and notebook keyboards related to key stiffness and typing force," *Ergonomics*, vol. 49, no. 10, pp. 996–1012, 2006.

[19] D. Cowles, "Keytop levelling mechanism," Feb. 21 1984. US Patent 4,433,225.

[20] O. Kamishima, "Keyboard switch for notebook type computer or the like," Apr. 13 1999. US Patent 5,894,117.

[21] E. Coleman, "Rocking switch actuator for a low force membrane contact switch," July 9 1985. US Patent 4,528,431.

[22] W. Davis and E. Sonderman, "Scan-controlled keyboard," Feb. 17 1982. EP Patent App. EP19,810,900,591.

[23] A. Chapweske, "The ps/2 keyboard interface." http://web.archive.org/web/20170831033351/ http://www.computer-engineering.org/ps2keyboard/, 2001. Accessed: 2017-08-31.

[24] U. I. Forum, *Universal Serial Bus (USB) Device Class Definition for Human Interface Devices (HID)*, 2001.

[25] I. Corporation, *Universal Host Controller Interface (UHCI) Design Guide*, 1996.

[26] H. Shimizu, "Measuring keyboard response delays by comparing keyboard and joystick inputs," *Behavior Research Methods*, vol. 34, no. 2, pp. 250–256, 2002.

[27] "Alps SKCL/SKCM Series Technical Specifications." http://web.archive.org/web/20160318052046/https://www.usbid.com/datasheets/usbid/2000/2000-q2/5454_31.pdf. Accessed: 2017-09-01.

[28] "Cherry MX Series Technical Specifications." http://web.archive.org/web/20170814022406/http://cherryamericas.com/product/mx-series-2/#84b4bc7a7a0396678. Accessed: 2017-09-01.

[29] M. Lipp, D. Gruss, M. Schwarz, D. Bidner, C. Maurice, and S. Mangard, "Practical keystroke timing attacks in sandboxed javascript," in *Proc. 22nd European Symp. on Research in Computer Security*, 2017.

[30] Microsoft, "Timers, timer resolution, and development of efficient code." http://web.archive.org/web/20170221051458/http://download.microsoft.com:80/download/3/0/2/3027D574-C433-412A-A8B6-5E0A75D5B 237/Timer-Resolution.docx, 2010-06-16. Accessed: 2017-02-21.

[31] A. Singh, *Mac OS X Internals: A Systems Approach*. Addison Wesley Professional, 2006. Section 7.4.1.1.

[32] "NO_HZ: Reducing Scheduling-Clock Ticks." http://web.archive.org/web/20170812022108/https://www.kernel.org/doc/Documentation/timers/NO_HZ.txt. Accessed: 2017-09-01.

[33] F. Collins, "Usb keystroke monitoring apparatus and method," Dec. 20 2007. US Patent App. 11/762,032.

[34] K. Subramanyam, C. E. Frank, and D. H. Galli, "Keyloggers: The overlooked threat to computer security," in *Proc. 1st Midstates Conf. for Undergraduate Research in Computer Science and Mathematics*, 2003.

[35] O. Zaitsev, "Skeleton keys: the purpose and applications of keyloggers," *Network Security*, vol. 2010, no. 10, pp. 12–17, 2010.

[36] T. Fiebig, J. Danisevskis, and M. Piekarska, "A metric for the evaluation and comparison of keylogger performance," in *Proc. 7th Usenix Conf. on Cyber Security Experimentation and Test*, pp. 7–7, USENIX Association, 2014.

[37] Y. Berger, A. Wool, and A. Yeredor, "Dictionary attacks using keyboard acoustic emanations," in *Proc. 13th ACM Conf. on Computer and communications security*, pp. 245–254, ACM, 2006.

[38] C. E. Shannon, "Prediction and entropy of printed english," *Bell Labs Technical Journal*, vol. 30, no. 1, pp. 50–64, 1951.

[39] T. Zhu, Q. Ma, S. Zhang, and Y. Liu, "Context-free attacks using keyboard acoustic emanations," in *Proc. ACM Conf. on Computer and Communications Security (CCS)*, pp. 453–464, ACM, 2014.

[40] D. Balzarotti, M. Cova, and G. Vigna, "Clearshot: Eavesdropping on keyboard input from video," in *Proc. IEEE Symp. on Security & Privacy (SP)*, pp. 170–183, IEEE, 2008.

[41] J.-B. Michel, Y. K. Shen, A. P. Aiden, A. Veres, M. K. Gray, J. P. Pickett, D. Hoiberg, D. Clancy, P. Norvig, J. Orwant, *et al.*, "Quantitative analysis of culture using millions of digitized books," *science*, vol. 331, no. 6014, pp. 176–182, 2011.

[42] A. Barisani and D. Bianco, "Sniffing keystrokes with lasers/voltmeters," *Proceedings of Black Hat USA*, 2009.

[43] T. A. Salthouse, "Perceptual, cognitive, and motoric aspects of transcription typing.," *Psychological bulletin*, vol. 99, no. 3, p. 303, 1986.

[44] T. A. Salthouse, "Effects of practice on a typing-like keying task," *Acta psychologica*, vol. 62, no. 2, pp. 189–198, 1986.

[45] B. Ritwik, S. FEng, J. S. Kang, and Y. Choi, "Keystroke patterns as prosody in digital writings: A case study with deceptive reviews and essays," in *Proc. Conf. on Empirical Methods in Natural Language Processing*, (Doha, Qatar), Association for Computational Linguistics, October 2014.

[46] A. M. Feit, D. Weir, and A. Oulasvirta, "How we type: Movement strategies and performance in everyday typing," in *Proc. ACM Conf. on Human Factors in Computing Systems (CHI)*, pp. 4262–4273, ACM, 2016.

[47] D. R. Gentner, "Keystroke timing in transcription typing," in *Cognitive aspects of skilled typewriting*, pp. 95–120, Springer, 1983.

[48] A. Neupane, M. L. Rahman, and N. Saxena, "Peep: Passively eavesdropping private input via brainwave signals," in *Proc. 21st Intl. Conf. on Financial Cryptography and Data Security (FC)*, pp. 227–246, IFCA, 2017.

[49] H. Wang, T. T.-T. Lai, and R. Roy Choudhury, "Mole: Motion leaks through smartwatch sensors," in *Proc. 21st Annual Intl. Conf. on Mobile Computing and Networking (MobiCom)*, pp. 155–166, ACM, 2015.

[50] C. Wang, X. Guo, Y. Wang, Y. Chen, and B. Liu, "Friend or foe?: Your wearable devices reveal your personal pin," in *Proc. 11th ACM Asia Conf. on Computer and Communications Security (ASIACCS)*, pp. 189–200, ACM, 2016.

[51] X. Liu, Z. Zhou, W. Diao, Z. Li, and K. Zhang, "When good becomes evil: Keystroke inference with smartwatch," in *Proc. 22nd ACM Conf.*

on Computer and Communications Security (CCS), pp. 1273–1285, ACM, 2015.

[52] A. Maiti, O. Armbruster, M. Jadliwala, and J. He, "Smartwatch-based keystroke inference attacks and context-aware protection mechanisms," in *Proc. 11th ACM Asia Conf. on Computer and Communications Security (ASIACCS)*, pp. 795–806, ACM, 2016.

[53] B. Chen, V. Yenamandra, and K. Srinivasan, "Tracking keystrokes using wireless signals," in *Proc. 13th Annual Intl. Conf. on Mobile Systems, Applications, and Services (MobiSys)*, pp. 31–44, ACM, 2015.

[54] D. Asonov and R. Agrawal, "Keyboard acoustic emanations," in *Proc. IEEE Symp. on Security & Privacy (SP)*, pp. 3–11, IEEE, 2004.

[55] T. Halevi and N. Saxena, "Keyboard acoustic side channel attacks: exploring realistic and security-sensitive scenarios," *Intl. Journal of Information Security*, vol. 14, no. 5, pp. 443–456, 2015.

[56] L. Zhuang, F. Zhou, and J. D. Tygar, "Keyboard acoustic emanations revisited," *ACM Trans. on Information and System Security (TISSEC)*, vol. 13, no. 1, p. 3, 2009.

[57] J. Liu, Y. Wang, G. Kar, Y. Chen, J. Yang, and M. Gruteser, "Snooping keystrokes with mm-level audio ranging on a single phone," in *Proc. 21st Annual Intl. Conf. on Mobile Computing and Networking (MobiCom)*, pp. 142–154, ACM, 2015.

[58] A. Compagno, M. Conti, D. Lain, and G. Tsudik, "Don't skype & type!: Acoustic eavesdropping in voice-over-ip," in *Proc. ACM on Asia Conf. on Computer and Communications Security (ASIACCS)*, pp. 703–715, ACM, 2017.

[59] P. Marquardt, A. Verma, H. Carter, and P. Traynor, "(sp) iphone: Decoding vibrations from nearby keyboards using mobile phone accelerometers," in *Proc. 18th ACM Conf. on Computer and Communications Security (CCS)*, pp. 551–562, ACM, 2011.

[60] K. Zhang and X. Wang, "Peeping tom in the neighborhood: Keystroke eavesdropping on multi-user systems," *analysis*, vol. 20, p. 23, 2009.

[61] S. Jana and V. Shmatikov, "Memento: Learning secrets from process footprints," in *Proc. IEEE Symp. on Security & Privacy (SP)*, pp. 143–157, IEEE, 2012.

[62] P. Vila and B. Kopf, "Loophole: Timing attacks on shared event loops in chrome," in *Proc. Usenix Security Symp.*, (Vancouver, BC), pp. 849–864, USENIX Association, 2017.

[63] J. T. Trostle, "Timing attacks against trusted path," in *Proc. IEEE Symp. on Security & Privacy (SP)*, pp. 125–134, IEEE, 1998.

[64] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *Proc. 16th ACM Conf. on Computer and Communications Security (CCS)*, pp. 199–212, ACM, 2009.

[65] D. Gruss, C. Maurice, K. Wagner, and S. Mangard, "Flush+ flush: a fast and stealthy cache attack," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 279–299, Springer, 2016.

[66] P. Pessl, D. Gruss, C. Maurice, M. Schwarz, and S. Mangard, "Drama: Exploiting dram addressing for cross-cpu attacks," in *Proc. 25th Usenix Security Symp.*, 2016.

[67] Y. Su, D. Genkin, D. Ranasinghe, and Y. Yarom, "USB snooping made easy: Crosstalk leakage attacks on USB hubs," in *Proc. Usenix Security Symp.*, (Vancouver, BC), pp. 1145–1161, USENIX Association, 2017.

[68] S. Chen, R. Wang, X. Wang, and K. Zhang, "Side-channel leaks in web applications: A reality today, a challenge tomorrow," in *Proc. IEEE Symp. on Security & Privacy (SP)*, pp. 191–206, IEEE, 2010.

[69] G. Zhang and S. Fischer-Hübner, "Timing attacks on pin input in voip networks (short paper)," in *Proc. Intl. Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 75–84, Springer, 2011.

[70] R. M. Chapman and H. R. Bragdon, "Evoked responses to numerical and non-numerical visual stimuli while problem solving," *Nature*, vol. 203, no. 4950, pp. 1155–1157, 1964.

[71] I. Martinovic, D. Davies, M. Frank, D. Perito, T. Ros, and D. Song, "On the feasibility of side-channel attacks with brain-computer interfaces," in *Proc. 21st Usenix Security Symp.*, USENIX Association, 2012.

[72] P. Shrestha and N. Saxena, "An offensive and defensive exposition of wearable computing," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, p. 92, 2017.

[73] W. Wang, A. X. Liu, M. Shahzad, K. Ling, and S. Lu, "Understanding and modeling of wifi signal based human activity recognition," in *Proc. 21st Annual Intl. Conf. on Mobile Computing and Networking*, pp. 65–76, ACM, 2015.

[74] K. Ali, A. X. Liu, W. Wang, and M. Shahzad, "Recognizing keystrokes using wifi devices," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 5, pp. 1175–1190, 2017.

[75] Q. Ge, Y. Yarom, D. Cock, and G. Heiser, "A survey of microarchitectural timing attacks and countermeasures on contemporary hardware," *Journal of Cryptographic Engineering*, pp. 1–27, 2016.

[76] A. Tannous, J. Trostle, M. Hassan, S. E. McLaughlin, and T. Jaeger, "New side channels targeted at passwords," in *Proc. Annual Computer Security Applications Conf. (ACSAC)*, pp. 45–54, IEEE, 2008.

[77] D. A. Osvik, A. Shamir, and E. Tromer, "Cache attacks and countermeasures: the case of aes," in *Proc. CryptographersŤrack at the RSA Conf.*, pp. 1–20, Springer, 2006.

[78] "Timing analysis is not a real-life threat to SSH secure shell users." http://web.archive.org/web/20010831024537/http://www.ssh.com/products/ssh/timing_analysis.cfm. Accessed: 2017-09-01.

[79] M. A. Hogye, C. T. Hughes, J. M. Sarfaty, and J. D. Wolf, "Analysis of the feasibility of keystroke timing attacks over ssh connections," *Research Project at University of Virginia*, 2001.

[80] G. Shah, A. Molina, M. Blaze, *et al.*, "Keyboards and covert channels.," in *Proc. Usenix Security Symp.*, vol. 15, 2006.

[81] J. P. Rosenfeld, M. Soskins, G. Bosh, and A. Ryan, "Simple, effective countermeasures to p300-based tests of detection of concealed information," *Psychophysiology*, vol. 41, no. 2, pp. 205–219, 2004.

[82] H. J. Chizeck and T. Bonaci, "Brain-computer interface anonymizer," Feb. 6 2014. US Patent App. 14/174,818.

[83] S. A. Anand and N. Saxena, "A sound for a sound: Mitigating acoustic side channel attacks on password keystrokes with active sounds," in *Proc. Intl. Conf. on Financial Cryptography and Data Security*, IFCA,IACR, 2016.

[84] R. Paavilainen, "Method and device for signal protection," Apr. 8 2008. US Patent 7,356,626.

[85] M. G. Kuhn and R. J. Anderson, "Soft tempest: Hidden data transmission using electromagnetic emanations," in *Proc. Intl. Workshop on Information Hiding*, pp. 124–142, Springer, 1998.

[86] R. Anderson and M. Kuhn, "Low cost countermeasures against compromising electromagnetic computer emanations," Apr. 13 2004. US Patent 6,721,423.

[87] "High Resolution Time Level 2." http://web.archive.org/web/20171017013909/https://www.w3.org/TR/hr-time/. Accessed: 2017-10-17.

[88] A. Askarov, D. Zhang, and A. C. Myers, "Predictive black-box mitigation of timing channels," in *Proc. 17th ACM Conf. on Computer and Communications Security (CCS)*, pp. 297–307, ACM, 2010.

[89] J. V. Monaco and C. C. Tappert, "Obfuscating keystroke time intervals to avoid identification and impersonation," in *Proc. Intl. Conf. on Biometrics (ICB)*, IEEE, 2016.

[90] K. Buza and P. B. Kis, "Towards privacy-aware keyboards," in *Proc. Intl. Conf. on Computer Recognition Systems*, pp. 140–147, Springer, Cham, 2017.

[91] M. Schwarz, C. Maurice, D. Gruss, and S. Mangard, "Fantastic timers and where to find them: High-resolution microarchitectural attacks in javascript," in *Proc. 21st Intl. Conf. on Financial Cryptography and Data Security (FC)*, p. 11, IFCA, 2017.

[92] C. Herley and P. van Oorschot, "Sok: Science, security and the elusive goal of security as a scientific pursuit," in *Proc. IEEE Symp. on Security & Privacy (SP)*, pp. 99–120, IEEE, 2017.

[93] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown," *arXiv preprint arXiv:1801.01207*, 2018.

[94] N. Yager and T. Dunstone, "The biometric menagerie," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 32, no. 2, pp. 220–230, 2010.

[95] P. M. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement.," *Journal of experimental psychology*, vol. 47, no. 6, p. 381, 1954.

[96] X. Bi, Y. Li, and S. Zhai, "Ffitts law: modeling finger touch with fitts' law," in *Proc. SIGCHI Conf. on Human Factors in Computing Systems*, pp. 1363–1372, ACM, 2013.

[97] J. D. Allen, *An analysis of pressure-based keystroke dynamics algorithms*. PhD thesis, Southern Methodist University, 2010.

[98] A. Morales, M. Falanga, J. Fierrez, C. Sansone, and J. Ortega-Garcia, "Keystroke dynamics recognition based on personal data: A comparative experimental evaluation implementing reproducible research," in *Proc. 7th IEEE Intl. Conf. on Biometrics Theory, Applications and Systems (BTAS)*, pp. 1–6, IEEE, 2015.

[99] N. Bakelman, J. V. Monaco, S.-H. Cha, and C. C. Tappert, "Keystroke biometric studies on password and numeric keypad input," in *Proc. Eu-*

ropean Intelligence and Security Informatics Conf. (EISIC), pp. 204–207, IEEE, 2013.

[100] Y. Li, B. Zhang, Y. Cao, S. Zhao, Y. Gao, and J. Liu, "Study on the beihang keystroke dynamics database," in *Proc. Intl. Joint Conf. on Biometrics (IJCB)*, pp. 1–5, IEEE, 2011.

[101] J. R. Montalvão Filho and E. O. Freire, "On the equalization of keystroke timing histograms," *Pattern Recognition Letters*, vol. 27, no. 13, pp. 1440–1446, 2006.

[102] Y. Sun, H. Ceker, and S. Upadhyaya, "Shared keystroke dataset for continuous authentication," in *Proc. 8th IEEE Intl. Workshop on Information Forensics and Security (WIFS)*, pp. 1–6, IEEE, 2016.

[103] E. P. Calot, "Keystroke dynamics keypress latency dataset." http://lsia.fi.uba.ar/pub/papers/kd-dataset/, 2015.

[104] N. Gonzalez and E. P. Calot, "Finite context modeling of keystroke dynamics in free text," in *Proc. Intl. Conf. of the Biometrics Special Interest Group (BIOSIG)*, pp. 1–5, IEEE, 2015.

[105] E. P. Calot and J. S. Ierache, "Multimodal biometric recording architecture for the exploitation of applications in the context of affective computing," in *Proc. 23rd Argentine Congress of Computer Science (CACIC), Innovation in Software Systems Workshop (WISS)*, UNLP, 2017.

[106] L. Bello, M. Bertacchini, C. Benitez, J. C. Pizzoni, and M. Cipriano, "Collection and publication of a fixed text keystroke dynamics dataset," in *Proc. XVI Congreso Argentino de Ciencias de la Computación*, 2010.

[107] E. Vural, J. Huang, D. Hou, and S. Schuckers, "Shared research dataset to support development of keystroke authentication," in *Proc. Intl. Joint Conf. on Biometrics (IJCB)*, pp. 1–8, IEEE, 2014.

[108] K. S. Killourhy and R. A. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics," in *Proc. IEEE/IFIP Intl. Conf. on Dependable Systems & Networks (DSN)*, pp. 125–134, IEEE, 2009.

[109] M. J. Coakley, J. V. Monaco, and C. C. Tappert, "Keystroke biometric studies with short numeric input on smartphones," in *Proc. IEEE 8th Intl. Conf. on Biometrics Theory, Applications and Systems (BTAS)*, pp. 1–6, IEEE, 2016.

[110] K. S. Killourhy and R. A. Maxion, "Free vs. transcribed text for keystroke-dynamics evaluations," in *Proc. Learning from Authoritative Security Experiment Results (LASER) Workshop*, pp. 1–8, ACM, 2012.

[111] R. Giot, M. El-Abed, and C. Rosenberger, "Greyc keystroke: a benchmark for keystroke dynamics biometric systems," in *Proc. IEEE 3rd Intl. Conf. on Biometrics: Theory, Applications, and Systems (BTAS)*, pp. 1–6, IEEE, 2009.

[112] S. Z. S. Idrus, E. Cherrier, C. Rosenberger, and P. Bours, "Soft biometrics database: A benchmark for keystroke dynamics biometric systems," in *Proc. Intl. Conf. of the Biometrics Special Interest Group (BIOSIG)*, pp. 1–8, IEEE, 2013.

[113] R. Giot, M. El-Abed, and C. Rosenberger, "Web-based benchmark for keystroke dynamics biometric systems: A statistical analysis," in *Proc. 8th Intl. Conf. on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, pp. 11–15, IEEE, 2012.

[114] D. Gunetti and C. Picardi, "Keystroke analysis of free text," *ACM Trans. on Information and System Security (TISSEC)*, vol. 8, no. 3, pp. 312–347, 2005.

[115] A. Morales, J. Fierrez, M. Gomez-Barrero, J. Ortega-Garcia, R. Daza, J. V. Monaco, J. Montalvão, J. Canuto, and A. George, "Kboc: Keystroke biometrics ongoing competition," in *Proc. IEEE 8th Intl. Conf. on Biometrics Theory, Applications and Systems (BTAS)*, pp. 1–6, IEEE, 2016.

[116] A. Morales, J. Fierrez, R. Tolosana, J. Ortega-Garcia, J. Galbally, M. Gomez-Barrero, A. Anjos, and S. Marcel, "Keystroke biometrics ongoing competition," *IEEE Access*, vol. 4, pp. 7736–7746, 2016.

[117] C. C. Loy, C. P. Lim, and W. K. Lai, "Pressure-based typing biometrics user authentication using the fuzzy artmap neural network," in *Proc. 12th Intl. Conf. on Neural Information Processing (ICONIP)*, pp. 647–652, Citeseer, 2005.

[118] C. C. Loy, W. K. Lai, and C. P. Lim, "Keystroke patterns classification using the artmap-fd neural network," in *Proc. 3rd Intl. Conf. on Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP)*, vol. 1, pp. 61–64, IEEE, 2007.

[119] M. Antal and L. Nemes, "The mobikey keystroke dynamics password database: Benchmark results," in *Software Engineering Perspectives and Application in Intelligent Systems*, pp. 35–46, Springer, 2016.

[120] M. Antal, L. Z. Szabó, and I. László, "Keystroke dynamics on android platform," *Procedia Technology*, vol. 19, pp. 820–826, 2015.

[121] M. Antal and L. Z. Szabó, "An evaluation of one-class and two-class classification algorithms for keystroke dynamics authentication on mobile devices," in *Proc. 20th Intl. Conf. on Control Systems and Computer Science (CSCS)*, pp. 343–350, IEEE, 2015.

[122] J. Roth, X. Liu, A. Ross, and D. Metaxas, "Biometric authentication via keystroke sound," in *Proc. Intl. Conf. on Biometrics (ICB)*, pp. 1–8, IEEE, 2013.

[123] J. Roth, X. Liu, and D. Metaxas, "On continuous user authentication via typing behavior," *IEEE Trans. on Image Processing*, vol. 23, no. 10, pp. 4611–4624, 2014.

[124] J. Roth, X. Liu, A. Ross, and D. Metaxas, "Investigating the discriminative power of keystroke sound," *IEEE Trans. on Information Forensics and Security*, vol. 10, no. 2, pp. 333–345, 2015.

[125] J. V. Monaco, G. Perez, C. C. Tappert, P. Bours, S. Mondal, S. Rajkumar, A. Morales, J. Fierrez, and J. Ortega-Garcia, "One-handed keystroke biometric identification competition," in *Proc. Intl. Conf. on Biometrics (ICB)*, pp. 58–64, IEEE, 2015.

[126] A. Pentel, "Predicting age and gender by keystroke dynamics and mouse patterns," in *Proc. 25th Conf. on User Modeling, Adaptation and Personalization (UMAP)*, pp. 381–385, ACM, 2017.

[127] K. Buza, "Person identification based on keystroke dynamics: Demo and open challenge.," in *Proc. 28th Intl. Conf. on Advanced Information Systems Engineering (CAiSE)*, pp. 161–168, 2016.

[128] M. El-Abed, M. Dafer, and R. El Khayat, "Rhu keystroke: A mobile-based benchmark for keystroke dynamics systems," in *Proc. Intl. Carnahan Conf. on Security Technology (ICCST)*, pp. 1–4, IEEE, 2014.

[129] G. Roffo, C. Giorgetta, R. Ferrario, W. Riviera, and M. Cristani, "Statistical analysis of personality and identity in chats using a keylogging platform," in *Proc. 16th Intl. Conf. on Multimodal Interaction*, ICMI '14, (New York, NY, USA), pp. 224–231, ACM, 2014.

[130] N. A. Laskaris, S. P. Zafeiriou, and L. Garefa, "Use of random time-intervals (rtis) generation for biometric verification," *Pattern Recognition*, vol. 42, no. 11, pp. 2787–2796, 2009.

[131] J. C. Stewart, J. V. Monaco, S.-H. Cha, and C. C. Tappert, "An investigation of keystroke and stylometry traits for authenticating online test takers," in *Proc. Intl. Joint Conf. on Biometrics (IJCB)*, pp. 1–7, IEEE, 2011.

[132] S. Koldijk, M. Sappelli, S. Verberne, M. A. Neerincx, and W. Kraaij, "The swell knowledge work dataset for stress and user modeling research," in *Proc. 16th Intl. Conf. on Multimodal Interaction*, pp. 291–298, ACM, 2014.

[133] M. Sappelli, S. Verberne, S. Koldijk, and W. Kraaij, "Collecting a dataset of information behaviour in context," in *Proc. 4th Workshop on Context-Awareness in Retrieval and Recommendation*, pp. 26–29, ACM, 2014.

[134] G. Ho, "Tapdynamics: strengthening user authentication on mobile phones with keystroke dynamics," tech. rep., Technical report, Technical report, Stanford University, 2014.

[135] Y. Uzun, K. Bicakci, and Y. Uzunay, "Could we distinguish child users from adults using keystroke dynamics?," *arXiv preprint arXiv:1511.05672*, 2015.

[136] C. C. Tappert, M. Villani, and S.-H. Cha, "Keystroke biometric identification and authentication on long-text input," *Behavioral biometrics for human identification: Intelligent applications*, pp. 342–367, 2009.

[137] K. Hempstalk, *Continuous typist verification using machine learning*. PhD thesis, The University of Waikato, 2009.

[138] K. Hempstalk, E. Frank, and I. Witten, "One-class classification by combining density and class probability estimation," *Machine Learning and Knowledge Discovery in Databases*, pp. 505–519, 2008.

[139] T. Holz, M. Engelberth, and F. Freiling, "Learning more about the underground economy: A case-study of keyloggers and dropzones," pp. 1–18, 2009.

[140] "Region-specific layouts." https://web.archive.org/web/20171030175546/https://deskthority.net/wiki/Region-specific_layouts. Accessed: 2017-10-30.

[141] A. Maas, C. Heather, C. T. Do, R. Brandman, D. Koller, and A. Ng, "Offering verified credentials in massive open online courses: Moocs and technology to advance learning and learning research (ubiquity symp.)," *Ubiquity*, vol. 2014, no. May, p. 2, 2014.

[142] R. Giot, B. Dorizzi, and C. Rosenberger, "A review on the public benchmark databases for static keystroke dynamics," *Computers & Security*, vol. 55, pp. 46–61, 2015.

APPENDIX

A. Keystroke Dataset Details

The dataset used to estimate information gain from a temporal side channel in Section III-C comes from a previous study that aimed to detect deceptive reviews and essays [45]. Each of the 1060 subjects were instructed to write genuine and fake reviews and then separately transcribe the responses within a web browser. For our analysis, only the free-text portion of the dataset was used to estimate information gain. We also applied the additional constraints of: discarding non-letter characters, discarding words that contain any modifier keys (e.g., Shift or Ctrl, to eliminate uppercase words and command inputs), and discarding words with any duration or latency >1 s. Only then were the bigrams within each word considered. This resulted in 1650±1046 (72±29 unique) bigrams per user. Our temporal information gain estimates are lower than those in [8], suggesting that context plays an import role in typing behavior (a phenomenon also noted by Salthouse [43]).

Some remarks on overlapping keystrokes are also warranted. Many keylogging attacks have operated under the assumption that keystrokes are non-overlapping and have an approximate duration of 100 ms (e.g., [39], [54], [56]–[59]). In practice, this is seldom the case except for some typists. In the same keystroke dataset described above, considering only alphanumeric keys (discarding modifier keys), more than 20% of all keystrokes are overlapping, i.e., $t_i^R > t_{i+1}^P$ for successive keystrokes. This also varies by typist, with some users having as high as 67% and 69% of all keystrokes overlapping for free and transcribed input, respectively. Only 20 users out of 1060, less than 2%, don't have any overlapping keystrokes.

B. Summary of Public Keystroke Datasets

We believe that access to public keystroke datasets can help facilitate future keylogging research, especially with respect temporal side channels. In general, real-world keylogging data is difficult to access and contains sensitive information [139]. Instead, we have identified over 30 public keystroke datasets from fields including biometrics, human-computer interaction, and affective computing. These are summarized in Table IV and characterized by the following attributes.

1) Locale: The locale reflects the country or region where the dataset was collected, which is either inferred from the respective reference or provided by the user agent string if available. The locale reflects both the physical layout (e.g., ISO vs ANSI) and logical layout (e.g., QWERTY vs DVORAK) of the keyboard (see Section II-A and [140]). For example, most US keyboards use ANSI/QWERTY; FR keyboards are dominated by ISO/AZERTY.

2) Typing mode: The *typing mode* is characterized by the acquisition context, which includes the instructions provided to the user and any keystroke input constraints imposed by the application and/or data collection environment. These broadly fall into three different categories:

- *Fixed*: The keystrokes exactly follow a relatively short predefined sequence. The character sequence is known beforehand and entered without errors or corrections. This includes the entry of passwords, phone numbers, and personal identification numbers (PINs) without making corrections.
- *Constrained*: The keystrokes roughly follow a predefined sequence, where typing errors and corrections are allowed. This includes case-insensitive passwords, passwords that were typed with corrections (i.e., including the Backspace key), and transcribed text. This mode of input is used by some massively open online course (MOOC) providers in which the student must copy several sentences for the purpose of keystroke dynamics-based verification [141].
- *Free*: The keystrokes do not follow a predefined sequence. The character sequence is unknown beforehand and usually consists of at least several sentences. The keystrokes collected as part of an essay question in an online exam would be considered long free-text.

The time intervals in freely-typed text tend to follow a much heaver tail than fixed-text. These larger latencies also tend to occur between certain keys, such as punctuation and the Space key. Note that *keystrokes* refer to the sequence of keys pressed, and the *character sequence* refers to the characters that are accepted by the application, for example what ultimately appears on screen in a text editor. The keystrokes may vary to produce the same character sequence when typing corrections are allowed.

3) Features: The most important features in a keystroke dataset are the key names and timings since these reflect the user's typing behavior and provide a ground truth to evaluate keylogger performance. The resolution and completeness of these features varies across datasets: in Table IV, datasets that contain key names but fail to distinguish between the location of modifier keys with multiple physical locations (e.g., LShift vs RShift) are marked as partial. Similarly for the timings, datasets that are missing either the key press or release timestamps are marked as partial.

Other features provide an opportunity to explore some of the side channel attacks described in Section IV. Several datasets contain audio/video recordings, which enable the possibility of acoustic and video side channels. Physiological features include hand motion, electrocardiography (ECG), and electroencephalogram (EEG), among others, which provide an opportunity to explore side channels from wearable devices, such as fitness trackers and smartwatches.

4) Attributes: Keylogging performance is subject to a number of environment variables related to the victim (e.g, age, gender, and handedness) and the platform (e.g., the keyboard model and timer resolution). Datasets that contain labels for these attributes are marked as such in Table IV. See [142] for a comprehensive review on environmental variables in keystroke acquisition.

5) Impostor Data: Many of the datasets in Table IV were collected with keystroke biometrics in mind and contain impostor data, that is a keystroke sequence recorded by a user purporting to be the genuine user.

TABLE IV

PUBLIC KEYSTROKE DATASETS. BR=BRAZIL, EST=ESTONIA, FI=FINLAND, FR=FRANCE, HU=HUNGARY, IT=ITALY, NE=NETHERLANDS, NO=NORWAY, RU=RUSSIA, SP=SPAIN, LA=LATIN AMERICA, TR=TURKEY, US(M)=UNITED STATES (MOBILE), -=UNKNOWN/NOT PROVIDED. ●=CONTAINS THE FEATURE; ◐=CONTAINS PART OF THE FEATURE; *no circle*=DOES NOT CONTAIN THE FEATURE.

			Typing Mode			**Features**						**Attributes**					
Name	**# Keystrokes**	**Locale**	**Fixed**	**Constrained**	**Free**	**Key timings**	**Key names**	**Pressure**	**Accelerometer**	**Audio/Video**	**Physiological**	**Age**	**Gender**	**Handedness**	**Platform**	**Impostor Data**	**Ref.**
allen	24k	US	✓			●	◐	●									[97]
atvs	85k	SP	✓			●										✓	[98]
bakelman	6.6k	US	✓			●	◐										[99]
banerjee	4.6M	-		✓	✓	●	◐										[45]
beihang	21k	-	✓			●	◐									✓	[100]
biochaves	32k	BR	✓		✓	◐	◐										[101]
buffalo	2.5M	US		✓	✓	●	●						✓		✓		[102]
calot	49k	LA			✓	◐											[103], [104]
calot-eeg	30k	LA		✓		●	●			●	●	✓	✓		✓		[105]
citefa	139k	US/SP/LA		✓		●	◐					✓	✓	✓	✓		[106]
clarkson	768k	US	✓	✓	✓	●	◐			●							[107]
cmu-password	224k	US	✓			●	◐										[108]
coakley	62k	USm	✓	✓		●	◐	●	●						✓		[109]
free-vs-trans	159k	US		✓	✓	●	◐										[110]
greyc	121k	FR	✓			●	◐										[111]
greyc-nislab	218k	FR/NO	✓			●	◐					✓	✓	✓			[112]
greyc-web	327k	US/FR	✓			●	◐					✓	✓		✓	✓	[113]
gunetti	605k	IT			✓	◐	◐									✓	[114]
how-we-type	37k	FI		✓		◐	●			●	●	✓	✓	✓			[46]
kboc	190k	SP	✓			●	◐									✓	[115], [116]
keystroke100	7.2k	-	✓			◐	◐	●									[117], [118]
mobikey	141k	USm	✓			●	◐	●	●						✓		[119]
mobile-password	30k	USm	✓			●	◐	●	●								[120], [121]
msu	49k	US		✓	✓	●	◐			●							[122]–[124]
ohkbic	440k	US			✓	●	◐					✓		✓			[125]
pentel	2.3M	EST/RU			✓	●	◐					✓					[126]
pic	14k	-		✓		●	◐										[127]
rhu	13k	USm	✓			●	◐					✓	✓				[128]
roffo	235k	IT			✓	◐											[129]
rti2014	55k	-	✓			◐											[130]
stewart	781k	US			✓	●	◐										[131]
swell-kw	249k	NE			✓	◐	◐			●	●	✓	✓	✓			[132], [133]
tapdynamics	8.4k	USm	✓			●	◐		●								[134]
uzun	11k	TR	✓			●	◐					✓	✓				[135]
villani	1.2M	US		✓	✓	●	●					✓	✓	✓	✓		[136]
waikato	132k	US			✓	●	◐										[137], [138]