ONLINE FIRST
13/5
2014
INTERNATIONAL JOURNAL OF
Information Security
Editors-in-Chief
Dieter Gollmann • Javier Lopez • Masahiro Mambo
Springer

Int. J. Inf. Secur.
DOI 10.1007/s10207-014-0264-7

REGULAR CONTRIBUTION

Keyboard acoustic side channel attacks: exploring realistic and security-sensitive scenarios

Tzipora Halevi · Nitesh Saxena

Abstract This research takes a closer look at keyboard acoustic emanations specifically for the purpose of eavesdropping over random passwords. In this scenario, dictionary and HMM language models are not applicable; the attacker can only utilize the raw acoustic information which has been recorded. This work investigates several existing signal processing techniques for this purpose and introduces a novel technique—*time–frequency decoding*—that improves the detection accuracy compared to previous techniques. It also carefully examines the effect of typing style—a crucial variable largely ignored by prior research—on the detection accuracy. The results show that using the same typing style (hunt and peck) for both training and decoding the data, the best case success rate for detecting correctly the typed key is 64 % per character. The results also show that changing the typing style, to touch typing, during the decoding stage reduces the success rate, but using the time–frequency technique, it is still possible to achieve a success rate of around 40 % per character. In these realistic scenarios, where the password is random, the approach described here can reduce the entropy of the search space by up to 57 % per character. This brings keyboard acoustic attack one step closer to a full-fledged vulnerability.

Keywords Keyboard acoustic emanations · Random passwords · Signal processing

T. Halevi (✉)
Computer Science and Engineering, Polytechnic Institute of New York University, Brooklyn, NY, USA
e-mail: thalevi@nyu.edu

N. Saxena
Computer and Information Sciences, University of Alabama at Birmingham, Birmingham, AL, USA
e-mail: saxena@cis.uab.edu

1 Introduction

The attacks based on acoustic emanations produced by electronic devices have been a known source of concern and present a threat to user privacy. Specifically, a few studies demonstrated that the seemingly conspicuous sounds resulting from keyboard typing can be used to learn information about the input data. Asonov and Agrawal [3] were the first to extract frequency features from the sound emanations of different keyboard clicks so as to identify the different keys used. Their work concluded that the physical plate beneath the keys causes each key to produce a different sound depending on its location on the plate (similar to hitting a drum at different locations). This makes keyboard typing vulnerable to eavesdropping attacks, in which similarities between clicks of the same key can be used to extract information about the keys pressed and the resulting data typed by the user.

Zhuang et al. [25,26] improved upon the attack of [3] by obviating the need for a labeled training recording. Instead, HMM English language-based model [14] was used on a 10-min typed English text for unsupervised training and labeling of the data [using neural networks and Mel Frequency Cepstrum Coefficients (MFCC)].

Berger et al. [6] further utilized dictionary attacks to decode eight letter or longer English words using correlation calculations. Their work showed that keys which are in close physical proximity on the keyboard typically have higher cross-correlation than farther ones.

1.1 Motivation

This paper takes a fresh look at keyboard acoustic attacks and aims to address some important aspects that prior work did not cover or fully explore. First, it systematically investi-

Published online: 20 September 2014

gates the possibility of eavesdropping over "random" textual passwords via keyboard acoustic emanations. Textual passwords are by far the most dominant means of user authentication deployed today, used in a variety of different contexts and applications. However, passwords suffer from several well-documented vulnerabilities [2,15,24]. One of the most prominent problems is that users often select "weak" passwords that can be easily guessed or are susceptible to small-space dictionary attacks (i.e., for a k-letter password, the size of the password space is much smaller than 26^k). In order to address this weakness, users are often instructed, and at times forced, to use random passwords [11,19]. These passwords possess relatively high bit entropy and employ random selection of characters. Therefore, in the realm of eavesdropping over a random password via keyboard acoustic emanations, a dictionary attack or an HMM language model is not useful and prior research is not applicable.[1] The first question this raises is *how feasible it is to retrieve random passwords by means of keyboard acoustic eavesdropping?*

In addition, this work examines the effect of typing style on key detection and eavesdropping ability. The hypothesis introduced in this work is that the typing style has a significant effect on the sound produced and can reduce the sound differences among clicks of different keys (as well as the similarities between separate clicks of the same key) which are due to the physical mechanics of the keyboard (as discovered in [3]). The reason typing style affects the audio emanations from the keys is that the finger touches the key from different angles as well as at different strengths. In addition, depending on the hitting angle, other fingers may touch/hit other keys, when using the touch typing style. Therefore, understanding the effect of the typing style is necessary to understanding keyboard eavesdropping attacks.

To the authors' knowledge, this is the first work that specifies the typing style employed in the experiments and analyzes/quantifies the impact of different typing styles. Reportedly, previous work has only used the "hunt and peck" or "search and peck" technique [1,23]. In this technique, as the name suggests, the typist finds and presses each key individually [21]. However, in real-life scenarios, many people employ "touch typing" [21]. Consequently, the second question asked in this work is *how much is the eavesdropping ability impacted by the variation in typing style, i.e., with respect to hunt and peck typing versus touch typing?*

The two questions posed above together motivate the research presented in this paper. This study starts by generating training data and then uses it to classify keys typed by making use of different signal processing techniques. The focus of this work is on eavesdropping random passwords. However, it is important to emphasize that in situations where a dictionary or HMM model may be used (e.g., while eavesdropping over English words or text), the techniques explored in this work are still useful. This work quantifies the ability of these techniques to provide information about the keys that can be combined with the language models wherever applicable.

[1] HMM model can still be useful for creating the training data, but not for the actual password guessing/decoding.

1.2 Summary of contributions

This work explores acoustic eavesdropping techniques under realistic and security-sensitive scenarios (different typing styles and random passwords). In doing so, it brings about several technical contributions. It provides an objective measure as to the degree of similarity between different presses of the same key and the ability to distinguish it from the presses of other keys. The work shows that the newly proposed time–frequency classification technique produces better results for decoding typed random passwords. This study presents two exhaustive search methods that significantly reduce the search space while considerably improving the detection capability over random search. Overall, it provides an objective indication as to the amount of data that eavesdropping can get about the keys (vs. language-based model attacks which depend on matching the typed data context), which is a prerequisite for the future design of appropriate defenses.

Paper organization: The remainder of this paper is organized as follows. The threat model is defined in Sect. 2. Section 3 describes the techniques used to detect pressed key and the performance of these techniques. Section 4 details the eavesdropping experiments conducted to test the effect of the different typing style on decoding random passwords. The performance of the techniques is presented in Sect. 5. A discussion of the results is provided in Sect. 6. Section 7 reviews some other work related to acoustic emanations and password attacks.

2 Threat model

The attack model used in this paper is very similar to the one considered by prior research on keyboard acoustic emanations [3,6,25,26]. Basically, the adversary is assumed to have installed a hidden audio listening device very close to the keyboard (or host computer) being used for user data input. A covert wireless "bug" or a PC microphone (perhaps a compromised microphone belonging to the host computer itself) is an example of such a listening device. The listening device can be programmed to record the acoustic emanations as the user types the data and transmits the recordings to another computer controlled by the adversary. This computer is then used for executing the signal processing and/or machine learning techniques in order to extract the input data corresponding to the recordings. In this study, random passwords consisting of lower-case English alphabets are considered. Since upper-case alphabet letters are typically typed in combination with the 'shift' key, this would require investigating overlapping sounds. However, since the goal of this research is quantifying detection capabilities of regularly typed keys, mixing both cases would affect the accuracy of the estimate of the detection probabilities of regularly typed keys. In addition, it is assumed that the adversary precisely knows the position of the password in the stream of all the data input by the user and recorded by the microphone.[2]

This attack examines in-depth the advantage which an adversary can obtain by comparing previously taken recordings of known data to new samples of data. In this scenario, the training and typed data will be captured with the same typing style. The attacker may also previously eavesdrop on an English text and use an HMM language-based model in order to recognize and label the typed keys, and use those samples to train the system. This is a realistic scenario, as the attacker can eavesdrop on the victim without his knowledge (using a disguised microphone) and even get a copy of the recorded data (such as, a company memo that is sent to other people or an email that is broadcast to multiple recipients). This scenario is emulated by using both training and testing data typed with the same typing style (this is done in two typing styles, as discussed in Sect. 4).

Another possibility is that the attacker itself gains access to the keyboard for a limited amount of time and uses the hunt and peck style to capture samples with the natural audio sounds of the keyboard, minimizing the effect of the individual typing style of the user. For this scenario, the training data are captured in a "mechanical" style (discussed in Sect. 4.1). These data can then be used to detect keyboard emanations of text recorded by the user at a later time (using a more natural touch typing style).

It is important to note that since an HMM language-based model or dictionary cannot be used for the attack and since passwords may be as short as six characters, producing some form of training data is necessary to eavesdrop over random passwords. Using the training data, audio information can be extracted *about the keyboard* and used later in the password guessing step.

Finally, it is assumed that the attacker has access to the device or a service that needs authentication for a limited amount of time. The attacker is usually allowed to make a certain number of password trials. A method of password exhaustive search that reduces significantly the overall search space, while increasing the probability of correct password detection is presented. The success probabilities of finding the correct password when the attacker is only allowed up to three trials is also provided in this study, as it is a common practice among many online services (especially banking web-sites) to lock out the owner's account after three failed attempts.

[2] Contextual or timing information may be used to determine this. As an example, the first keyboard input a user may provide every morning, while logging to her work computer, would usually be a password.

Attack setup and tools: Throughout the experiments presented in this work, a standard Lenovo keyboard (model JME7053 English) was used for typing, a Logitech USB PC microphone, the Recordpad (V.3.0.3) and Matlab software.

3 Detection techniques

Following is an overview of the techniques explored in this work for the detection of individual keys/characters pressed on the keyboard.

3.1 Determining key press signal

Keyboard acoustic signals have two distinct regions: push (also referred to as press) and release (Fig. 1), as demonstrated in [3]. The push region relates to the period where the finger touches the keyboard, while the release is the sound generated when the key is released. However, experiments conducted in this study showed that depending on the force sustained while pressing the key, both the push and the release have between 1 and 3 distinct peaks. To detect the optimal press period, the correlation of different signals generated by the same keys was compared. Initially, a regions of 3 ms long were used, and two cases were compared: choosing the first peak in the push region versus using the most pronounced one, with the latter producing better results. To detect the optimal period length, different lengths of regions were examined. The experiments showed that using regions

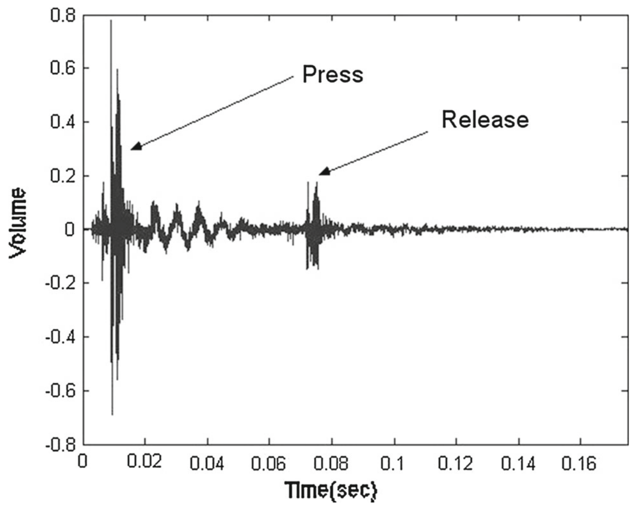


Fig. 1 Acoustic signal of a single key

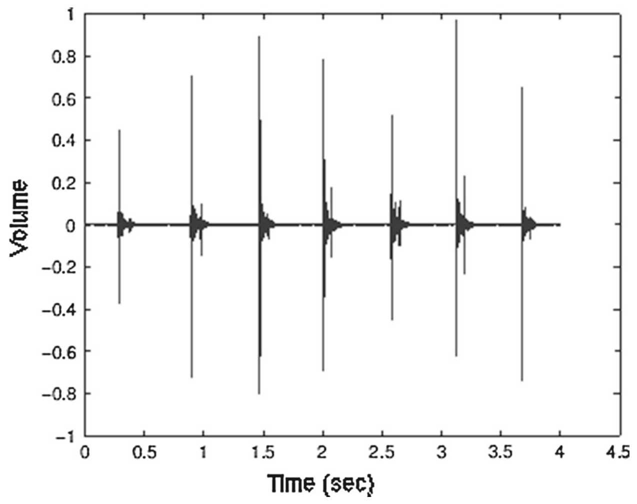


Fig. 2 Recording of multiple keys

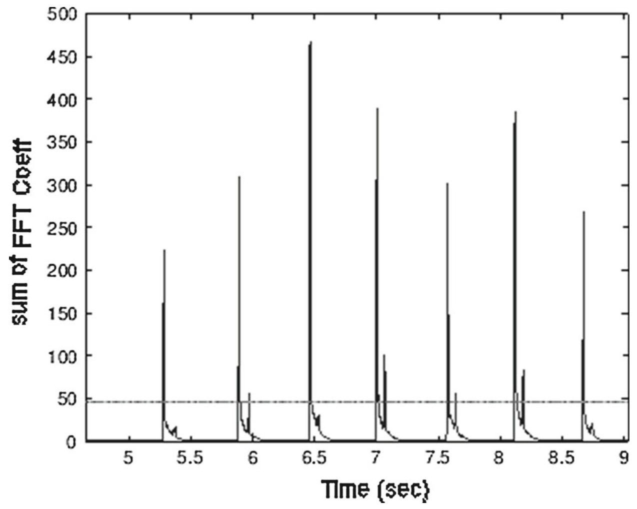


Fig. 3 Sum of FFT coefficients

of 50 ms (which started from the beginning of the first press) produced the best results.

Detecting key press regions: When recording the signals, pinpointing the exact beginning of the press signal was required. The signals were recorded with a sampling frequency of 44.1 kHz. To detect the beginning of each press, the fast Fourier transform (FFT) coefficients of the signal were calculated (utilizing the Matlab "specgram" command) using a window size of 440 samples. The FFT coefficients in the range of 0.4–22 kHz were summed-up, and a threshold was used to detect the beginning of each keypress (Figs. 2, 3).

To detect the key release region, the area following the push region (first 50 ms section of the signal) was examined. Since the release is less pronounced, a smaller window of 88 samples was used and the same process as the one for detecting the press signal was repeated.

3.2 Existing techniques

To optimize the signal decoding, a few techniques were evaluated. These included the Dynamic Time Warping (DTW) [16], which is an algorithm used to measure similarities between sequences that may vary in time. The technique has been widely used in speech recognition with very good results, improving significantly speech detection relative to previously used processing methods. The results of this technique are compared to previously used key detection techniques, both time and frequency based. This work further introduces a novel time–frequency-based classification technique which combines both the time domain data as well as the frequency features for improved detection results.

Dynamic time warping: Dynamic Time Warping (DTW) is an algorithm that outputs a similarity measurement between two sequences that may be compressed (or stretched) in time. The algorithm has been used in automatic speech recognition, to cope with different speaking speeds and provided significant improvement relative to the previous processing methods used for this application.

To evaluate the algorithm, the simple distance measure between each two elements in the signal vectors was used. The DTW function was implemented using C source code for a Matlab executable (MEX), which resulted in a faster overall running time compared to regular Matlab code.

Different signal normalization methods were compared, based on energy, amplitude, mutual joint distribution [12], and no normalization. The energy-based normalization was found to produce the best results. To detect the optimal regions of the signals, using only the push, only the release or the mean of both was tried, with the latter producing the best results.

Letter Data Set A dataset is created for each letter. Each Letter Data Set is made up of n samples that are typed for the corresponding alphabet letter.
Distance between test sample and Letter Data Set The distance between each test sample and each Letter Data Set is defined as the average of the distances between the test sample and each training sample belonging to the Letter Data Set.

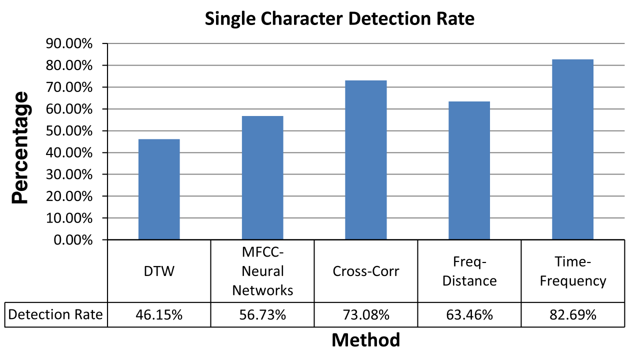
The DTW technique produces a distance measure between each two signals. To match each test sample with an alphabet key, two possible methods were compared. In the first case, the closest training sample to the test sample from the whole training data is chosen, and the alphabet letter which was typed to produce the training sample is picked. In the second case, the average distance between the test character and each alphabet key is calculated. This is done by getting the mean of the distance measurement between the test character and the Letter Data Set of each alphabet key. The best match is picked to be the alphabet key which has the smallest average distance to the test key. Using the mean value for each training alphabet key was found to produce better results, since it minimizes the noise contribution of each single test instance. Therefore, each test sample is assigned the alphabet letter which corresponds to the training Letter Data Set that produces the smallest distance.

Cross-correlation: The cross-correlation (denoted X-Corr) was performed between the recorded signals as done in [6]. The signals were normalized according to their energy, the X-Corr was calculated for the press and release regions, and the mean value of both was used. For each alphabet key, the cross-correlation between its Letter Data Set and each test sample was calculated, and the average of these measurements was obtained and used as a similarity measurement for each key. The matching alphabet letter was chosen as the one with the highest similarity.

Frequency-based distance measure: The frequency-domain features-based distance measure (denoted Freq-Dist) was performed similarly to the method described in [6]. The frequency spectrum was produced by calculating the FFT coefficients for both the press and release regions. Different bands of frequencies were examined, and the best results were achieved when using the coefficients in the 0.4–22 kHz. For each of the signals, the coefficients were normalized to one. The frequency-based distance between each two signals was obtained by calculating the Euclidean difference between the features for the press and release parts, with their average producing one distance measure. The distance between each test sample and each alphabet Letter Data Set was calculated, and the letter with the smallest distance was chosen.

Frequency features and neural networks: This frequency-domain features-based technique, described in [3], was implemented and tested on the test data. To detect the most active 3-ms window corresponding to the press region, the algorithm described in Sect. 3.1 was used, with a window size of 3 ms (132 samples). The signal spectrum was calculated, and the FFT coefficients were summarized over the 0.4–22 kHz, using a threshold to detect when the peak press began.

After creating the frequency-domain features, the Matlab feed-forward neural network was used to classify the keys. However, it was not possible to reproduce the results described in [3]. The experiments indicated that this technique was worse than any of the other methods tested. It is important to note that previous research [25,26] also could not achieve results similar to the original ones described. The difference in the findings could be due to different reasons (the original data used for those tests are not currently available, so an exact determination is not possible). The keyboard used for the tests in this work is different from the one used



	DTW	MFCC-Neural Networks	Cross-Corr	Freq-Distance	Time-Frequency
Detection Rate	46.15%	56.73%	73.08%	63.46%	82.69%

Fig. 4 Single character detection. Proposed time–frequency algorithm provides the highest detection rate, followed by the cross-correlation and the rest of the algorithms

in the experiments of [3] (which may have produced acoustic emanations with higher volume or more pronounced characteristics). Another difference could result from the fact that in this work, the Matlab neural network was used, while the original research used Java neural network. Also, the authors of [3] did not specify how to choose the press and release regions. In this work, an automatic method was chosen, but other methods (based on visual examination of the signal) may be used. However, such methods will be less efficient and not quite feasible for an eavesdropper using real-time data.

A second frequency-domain features-based technique was implemented, using MFCC features as input to neural networks, as described in [26]. In this work, a 10-ms windows and an 2.5 window step size were chosen, computing 13 MFCC per window, and examined a total of 40 ms of each press. As per the original implementation, in this work, the Matlab's *newpnn*() function was utilized for creating the neural network.

3.3 Performance

The aforementioned techniques were evaluated on the initial dataset, which was taken with the hunt and peck style (Sect. 4.1). For each sample in the initial dataset, the rest of the samples in the dataset (excluding the test sample) were used as training data. Due to the relatively high computation requirements for the DTW algorithm, only four instances per alphabet letter were used for training. The key detection rate (out of 26 alphabet letters) was tested using each technique. The cross-correlation technique was found to provide the best results, with a single key detection rate of 73 %. The detection results can be found in Fig. 4. All the techniques in the table significantly raise the decoding rates over random guess (which is less then 4 %).

The DTW technique gives lower detection results compared to the correlation algorithm, indicating that the signals do not vary much in time. Therefore, attempting to warp, the signals reduces the differences between clicks of different keys as it removes parts of the signals that provide useful information needed for more accurate decoding.

3.4 New technique: time–frequency classification

3.4.1 Definitions

The following notations are used throughout this section:

- $\mathbf{x}$ and $\mathbf{y}$—vectors of real numbers of length N that correspond to the signal audio measurement for a typed key.
- $\mathbf{x}(press)$ and $\mathbf{x}(release)$—vectors of real numbers as above (N is a parameter) for the press signal and release signal. $\mathbf{x}$ is the audio signal of the key sample that is being decoded.
- $\mathbf{y}_i$—an audio training sample received when typing the i'th alphabet letter. For each alphabet letter, a Letter Data Set of n training samples is generated. Each sample has one press signal and one release signal.
- $\mathbf{y}_i(j, press)$ and $\mathbf{y}_i(j, release)$—the j'th training sample press and release signals ($0 < j \leq n$)

3.4.2 Technique

In the time–frequency classification method (denoted Time–Frq), both the correlation calculation and the frequency-based calculations are combined to choose the best matching letter for each training letter. To achieve this, the frequency distance measure DF (Sect. 3.2) and X-Corr similarity measurement C are calculated for each instance.

To combine both elements, the correlation-based distance is defined as: $DC = 1 - C$ (so both DF and DC are ascending).

A few methods were examined to combine both metrics, including picking the minimum of each value ($\min(DC, DF)$) and the average of the two values (DC, DF). Another option explored was using (DF, DC) as a point on a 2D space and calculating the geometric mean (the Euclidean distance from zero). The latter option was found to provide the best results. The reason is that typically the frequency-based distance was smaller than the time-based distance (and was also less accurate when used by itself). Using the overall distance minimizes the contribution of the lower value (the frequency-based distance in this case). However, when the frequency-based distance is relatively large between both data vectors, this measurement helps distinguish correctly between two signals that belong to two different keys.

Therefore, the Euclidean distance is used as the distance measure for classifying each key (denoted as DTF) in this technique. The time–frequency similarity measure is further defined as $S_{TF} = 1 - DTF$, and the alphabet letter with the highest similarity to each test sample is chosen. The algorithm is further described in Algorithm 1 in the "Appendix 9".

Using the time–frequency classification technique, an increased probability of 83 % is achieved for the training data. Therefore, these results imply that both the frequency and the time data can be used together to produce better results.

4 Effect of typing styles

4.1 Datasets

To examine the effect of typing style on detection of pressed keys, three datasets were created. The first dataset was created for data training. The other two were testing datasets, each typed with a different typing style.

Straw man approach: The first scenario involves typing each letter multiple times (continuously) always using the same finger before moving to the next letter. A few seconds are allowed before typing the next letter (similar to the technique used in [3]). This causes the finger to hit the key from a vertical position in each case. The benefit of using this technique is that it ensures virtually no overlap of keyboard acoustic sounds. It also enables typing each letter using approximately the same force and hitting the keys from the same angle, resulting in a relatively similar sound for multiple clicks of the same key. Overall, this technique minimizes noise or overlap sounds during the key press and maximized the contribution of the keys hitting the underlying plate. Since the plate acts like a "drum", it produces the emanated audio sound [3] related to its position on the plate.

This technique can therefore be used to train the system by an attacker (not the original typist) who is trying to extract audio emanations which are due to the physical structure of the keyboard.

The above technique was used to capture ten signal recordings for each key of the alphabet letters as the initial data.

Hunt and peck typing: In the second scenario, random passwords are typed using the hunt and peck style. This case differs from the first case since consecutive letters are different from each other. This causes the finger to hit the key from possibly different angles (depending on which key was typed earlier). For this test, random passwords of six character each were generated (since the characters are chosen randomly, the data could be divided into any password size). Since six character is still the minimum size of password one can chose for many sites today, this still provides a realistic scenario where the attacker has the highest probability of guessing the password. A total of 25 different such random passwords were generated, and each password was typed three times consecutively. These data are refereed to as the "Test Hunt and Peck data" in the rest of the paper.

Touch typing: In the third scenario, the same password list is typed—as in the Hunt and Peck case—using the touch typing technique. In this scenario, each key has its own designated finger and the rest of the fingers may possibly touch the keyboard while typing (depending on the hands' movement). This typing technique is very popular among users. However, this typing style does affect the acoustic emanations as the keys are hit from different angles, depending both on the finger used as well as the hand position during the typing of each key (which depends on the previous letters typed). These data are referred to as the "Test Touch Typing data".

4.2 Typing style and signal correlation

To measure the effect of typing style on signal similarity, the maximum cross-correlation between instances of the keys in the test data with each Letter Data Set of the training data is calculated.

The training data included 10 training samples using the straw man typed dataset (Sect. 4.1).

Straw man typing: In this case, the aforementioned data were used as test data itself. For each sample, the maximum cross-correlation is calculated with each of the other instances taken with the same key (termed *matching key*). The mean of all the values for each key is then calculated. This was repeated for both the press and release part of the signal. These values are marked as $\mathsf{CorrMatchPrs}(i)$ and $\mathsf{CorrMatchRls}(i)$ for each sample i of the data.

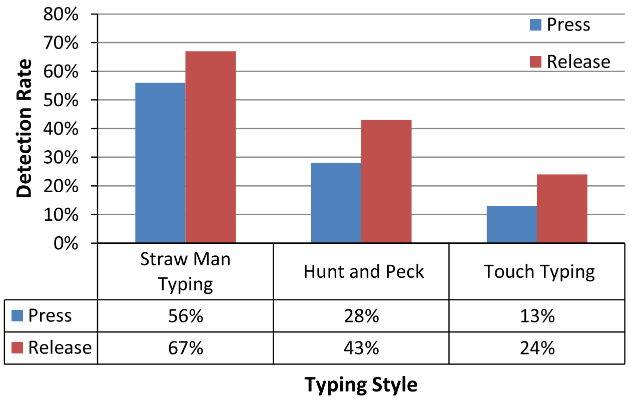
This calculation was repeated for each test sample with the Letter Data Sets of the rest of the keys (termed *non-matching keys*). The resulting values are marked as $\mathsf{CorrPrs}(i, j)$ and $\mathsf{CorrRls}(i, j)$, where j is the alphabet letter that corresponds to each Letter Data Set. For each tested sample, the highest value of the 25 values is chosen, which shows the correlation to the most likely key to be chosen as a match to the original sample. This value is marked as $\mathsf{CorrNonMatchPress}(i)$ and $\mathsf{CorrNonMatchRls}(i)$ for each sample i of the data. Therefore,

$$\mathsf{CorrNonMatchPress}(i) = \arg\max_{j, j \neq i}(\mathsf{CorrPrs}(i, j)) \tag{1}$$

At this point, the correlations of the press and release samples were compared. If the highest correlation for the sample belongs to the Letter Data Set of the matching alphabet key, i.e.,

$$\mathsf{CorrMatchPrs}(i) > \mathsf{CorrNonMatchPrs}(i), \tag{2}$$

the press part of the sample is marked as a match correlation. The same is repeated for the release part of the signal. The overall match probability is calculated as the number of keys found to match (i.e., being best correlated to the samples of the corresponding typed letter in the training data) divided by the total number of samples n. Therefore,



	Straw Man Typing	Hunt and Peck	Touch Typing
🟦 Press	56%	28%	13%
🟥 Release	67%	43%	24%

Fig. 5 Probability of keys matching the training data with typing style variation

$$\mathsf{PMatchPress} = \frac{\sum r}{n}; \tag{3}$$

for all r such that:

$$\mathsf{CorrMatchPrs}(r) > \mathsf{CorrNonMatchPrs}(r),$$

The release match probability is calculated in the same way using the release correlation values. Using this calculation, for the straw man dataset, 56 % of the press signals and 67 % of the release signals best matched their corresponding typed letter.

Hunt and peck typing: The correlation calculations were repeated between the hunt and peck passwords test dataset and the straw man initial dataset. In this case, the match probability was reduced to 28 % for the press and 43 % for the release samples.

Therefore, it was observed that the percentage of signals that are best correlated with the training data belonging to the matching letter is significantly reduced. Consequently, when the typing style changed slightly it becomes more likely to choose the wrong key as the best matching key to the new sample.

Touch typing: For the data typed with the touch typing style, performing the same analysis produced lower matching results. In this case, the probability of each instance matching the correct letter in the training data was reduced to 13 % for the press part of the signal and 24 % for the release.

A summary of the results is presented in Fig. 5. In conclusion, it was observed that the maximum correlation between instances of the same key reduces when the typing style changes. On the other hand, the correlation to instances taken with other keys increases which makes it hard to detect correctly the key. This confirms the original hypothesis that typing style has a strong effect on the similarity of same key audio signals and the ability to distinguish them from other keys.

5 Password detection

Out of the five techniques explored in Sect. 3, the cross-correlation (X-Corr) and time–frequency classification (Time–Frq) techniques yielded higher detection accuracy. In this section, the advantage that an attacker can get by using these two techniques to eavesdrop over random passwords is investigated.

The performance of these techniques was examined, and the detection rates were compared for random passwords typed with both the hunt and peck and the touch typing styles.

To improve the accuracy of the detection, ten instances of training data for each alphabet key are used for each of the data groups. This reduces the effect of noise on the results (relatively to the four instances used in Sect. 4.1).

For each alphabet key letter in the training data, the similarity measure—max correlation for cross-correlation and S_{TF} similarity for time–frequency classification—is calculated. For each technique, the best matching letter was chosen as the one with the highest similarity measurement.

5.1 Initial dataset, straw man approach

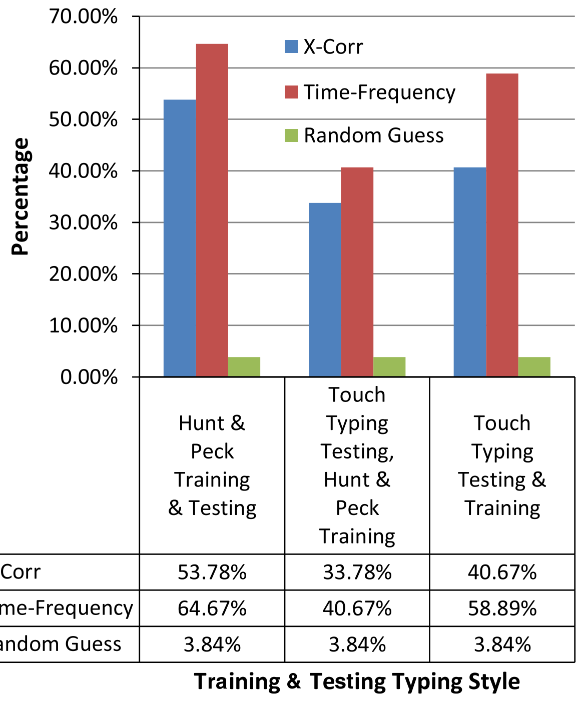
The initial data are typed with the straw man approach (as discussed in Sect. 4.1). Each instance in the dataset is utilized as a test instance, with the rest of the dataset used as the training data (excluding the test instance).

As a result, the cross-correlation statistics calculated using this technique resulted in 83 % accuracy rate per key (since now ten keys per training were used per key, this raised the result up from 73 % when only four training instances were used). When using the time–frequency-based classification, the results further improved (to 89 %). This leads to the conclusion that when the typing is repetitive, the underlying physical characteristics of the keyboard have strong effect on the acoustic emanations and the ability to eavesdrop is relatively high.

5.2 Test data, hunt and peck style

For this dataset, the similarity measures are calculated using the initial data (straw man approach dataset) as the training data and the hunt and peck dataset as the test dataset.

In this scenario, the cross-correlation performance was reduced to a 54 % accuracy rate per key. Utilizing the time–frequency technique improved the detection rate to 65 % per character (see Fig. 6). These results suggest that the angle at which the finger hits the key affects the acoustic signal emanated and reduces the detection accuracy compared to typing the same key continuously.



	Hunt & Peck Training & Testing	Touch Typing Testing, Hunt & Peck Training	Touch Typing Testing & Training
🟦 X-Corr	53.78%	33.78%	40.67%
🟥 Time-Frequency	64.67%	40.67%	58.89%
🟩 Random Guess	3.84%	3.84%	3.84%

Fig. 6 Single character detection rates, best character guess

5.3 Test data, touch typing style

In this scenario, the test passwords are typed with the touch typing style (using the straw man dataset as the training data). Utilizing the cross-correlation technique for key detection, the accuracy rate was reduced to 34 %. For the time–frequency-based classification, the rate of detection per correct character has increased to 41 % (see Fig. 6).

5.4 Best guesses search

To raise the detection rate, a list of additional possible key matches is created that can be checked against the recorded passwords. This is implemented by creating a list of keys having the highest max correlation and a list of keys with the lowest TF distance from the test character. When examining the ordered list of highest matching alphabet letters, the probability of the key matching each of the letters was found to reduce significantly after the fifth letter.

Therefore, a **"Best Guesses Search"** method was implemented—in which for each typed character, a list of the five best matching keys is created. Using this list, the probability of a correct detection for those five keys is calculated. Utilizing the correlation-based technique, the probability of each character to be in the list of the top five keys increased to 79 % for the hunt and peck data. For the touch typing data, in contrast, the probability that the key is in the first five choices was found to be 64 %.

For the time–frequency-based classification, the probability of each character to be in the list of the top five keys increased to 88 % (for the hunt and peck data). For the touch typing data, the rate increased to 75 %. Those results (for single key and five key guesses) are summarized in Figs. 6 and 11, respectively. The detection rates for the key being in the two best guesses are provided as well in Fig. 12.

5.5 Training, touch typing style

In this scenario, the attacker first eavesdrops over a user typing continuous text. He records the text and uses language model tools to detect the keys pressed. Then, when the user types his password (testing phase), the attacker uses the previous recordings he has as training data to decode the characters typed.

To perform this test, the training data were created by typing each letter continuously and recording the audio signal, using touch typing. A comparison between decoding the password data using both the cross-correlation and the time–frequency method is presented, using the continuous recorded training data.

The tests conducted showed that as expected, the password decoding has significantly improved in this case (compared to training with hunt and peck style data). The results are summarized in Fig. 6.

5.6 Algorithm search entropy

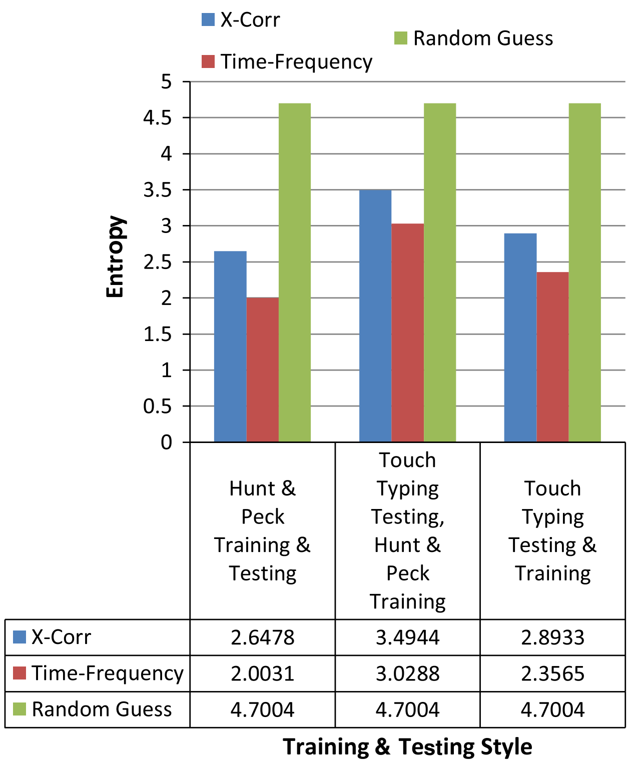
To add a more general measurement of the improvement in password search by the time–frequency decoding algorithm search, the entropy of the algorithm is calculated according to the following equation:

$$Entropy = -\sum_{i=1}^{26} p_i \times \log_2 p_i \tag{4}$$

The results of the improvement in entropy search are provided in Fig. 7. As can be seen in the table, the time–frequency algorithm reduces the entropy significantly. For the hunt & peck typing style, which was the style used in previous research papers, the time–frequency algorithm reduces the entropy by 57 %, followed by a reduction of 40 % for the touch typing style (where both training and testing are in touch typing style) and followed by a reduction of 36 % for the case where the training and testing are in different typing style.

5.6.1 Search optimization

In [22], Charvillon et. al. suggested a graphical method of describing how to find the next optimal step when a few



	Hunt & Peck Training & Testing	Touch Typing Testing, Hunt & Peck Training	Touch Typing Testing & Training
🟦 X-Corr	2.6478	3.4944	2.8933
🟥 Time-Frequency	2.0031	3.0288	2.3565
🟩 Random Guess	4.7004	4.7004	4.7004

Fig. 7 Detection Algorithm Search Entropy. The time–frequency algorithm reduces the entropy by 36–57 %, depending on the training and typing style combination

lists are combined. For two lists, a 2D array of rectangles is created. The width of each row in the array is proportional to the probability of the next key in the first list, and the length of each column is proportional to the probability of the next key in the second list. Choosing which list to advance on is determined by choosing the adjacent rectangle with the highest probability (therefore choosing the rectangle with the largest area). The same algorithm can be used similarly for combining *n* lists.

For the algorithm presented in this paper, this method can be adopted when performing the search and using the best five guesses in the sorted guess list. However, after the best five guesses, the probability tapers off (and is affected significantly by noise due to the small probabilities). Therefore, to conduct an efficient search, the algorithm should be performed on the best five guesses. The optimization according to [22] can be described graphically by creating an *n*-dimension cube with five division in each dimension. For each *i*'th division in each dimension, the length would correspond to the probability of the *i*'th guess. After the first five guesses, the search should use the rest of the guesses in the sorted order (i.e., first going through all of the combinations that include the sixth-best guesses for each character, then the seventh, etc). This would allow optimization of the search. This analysis is based on the probability calculations, and this method has not been implemented or tested directly on the database used in this paper.

The probabilities of the first five guesses for each digit are provided in Fig. 8.

5.7 Password decoding

To evaluate the advantage that an eavesdropper can achieve by using an exhaustive search to detect an n-character password (i.e., by making use of a certain number of trials), a new search method is introduced in this work. While a brute-force attack on the entire password space would take $26^n \simeq 2^{4.7\times n}$ trials, an alternative method, the 'Best Guesses Search', is shown to reduce significantly the computing complexity and speed-up the attack. Therefore, it can be used when limited-time access is available to a device that requires password authentication. The success probabilities of finding the correct password are provided when the attacker is only allowed up to three trials (a common scenario).

In the Best Guesses Search, for each character, the closest five keys are chosen. Therefore, the number of tests is reduced to:

$$N(n) = 5^n \simeq 2^{2.3\times n} \tag{5}$$

This yields a probability of detecting the full password:

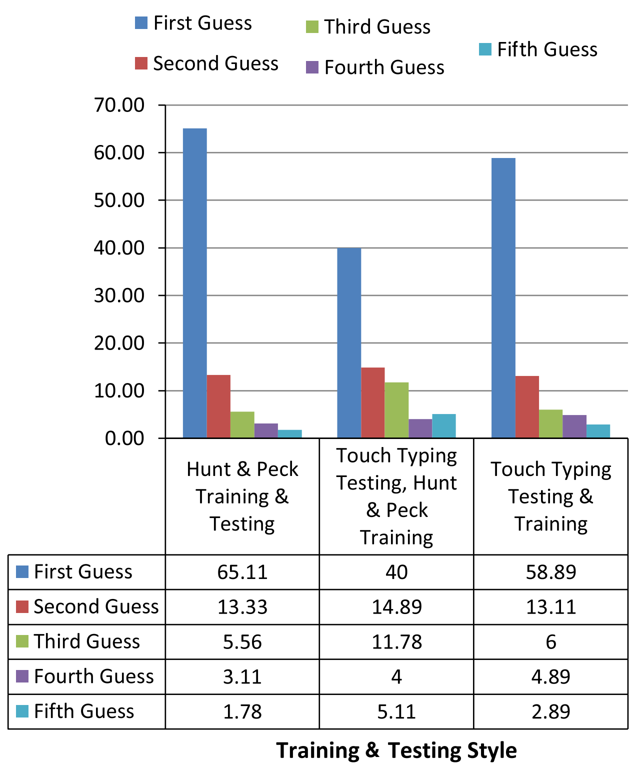
$$Pr_{PasswordDetection}(n) = (\mathsf{Pchar}_5)^n \tag{6}$$

Pchar_5 is the probability that a char matches one of the five guesses.

Overall, this means that the attack can cut down the entropy of the password from $n \times log(26)$ to $n \times log(5)$, about square root of the original search size space.

The accuracies of the cross-correlation and time–frequency classification techniques are compared when detecting $n - character$ passwords for the Best Guesses Search as well as when performing a small number of trials. The Best Guesses Search detection probability was tested for six character passwords (which includes $5^6 = 15{,}625$ trials). This test verified that it indeed matches the calculations using Eq. 6.

All of the detection rates are summarized in Table 1. The detection rates, for the case involving 1 to 3 trials, are obtained directly from the detection rates corresponding to the best matching single character listed in Figs. 6, 11 and 12. It is important to emphasize that for the Best Guesses Search, even when the passwords are typed with touch typing and the training uses hunt & peck typed data, the performance is still considerably better than a random guess attack (which would produce on average 0.005 % success rate for the size of the tested search space).



	Hunt & Peck Training & Testing	Touch Typing Testing, Hunt & Peck Training	Touch Typing Testing & Training
🟦 First Guess	65.11	40	58.89
🟥 Second Guess	13.33	14.89	13.11
🟩 Third Guess	5.56	11.78	6
🟪 Fourth Guess	3.11	4	4.89
🟦 Fifth Guess	1.78	5.11	2.89

Fig. 8 Probability of best five guesses. The probability is significantly reduced from the first guess to the second guess and tapers off continuously in the sorted guess list

Table 1 Six character password detection rates

Method →	Exhaustive search						Random guess (%)
Training →	Cross-correlation			Time–frequency			
Testing →	Hunt & peck		Touch	Hunt & peck		Touch	
# Trials ↓	H&P (%)	Touch (%)	Touch (%)	H&P (%)	Touch (%)	Touch (%)	
1	2.42	0.15	1.44	7.31	0.38	4.17	3.24E−07
2	2.92	0.19	1.80	8.87	0.59	5.14	6.47E−07
3	3.42	0.24	2.16	10.43	0.80	6.12	9.71E−07
15,625	24.00	5.33	22.67	42.67	21.33	34.67	0.0051

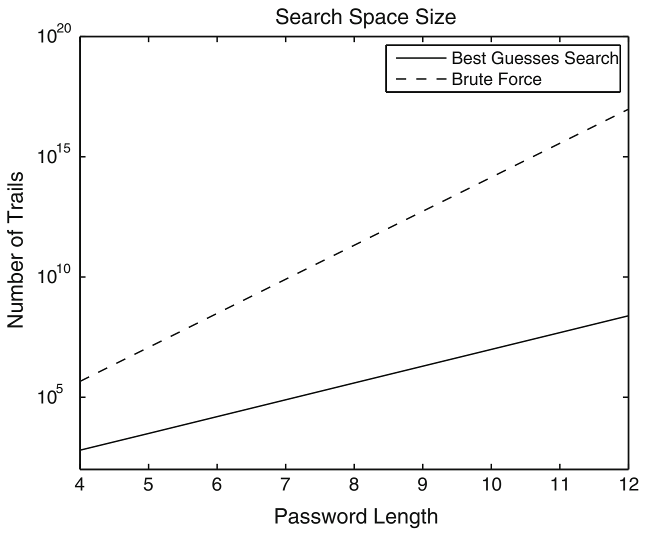


Fig. 9 Best guesses search space size

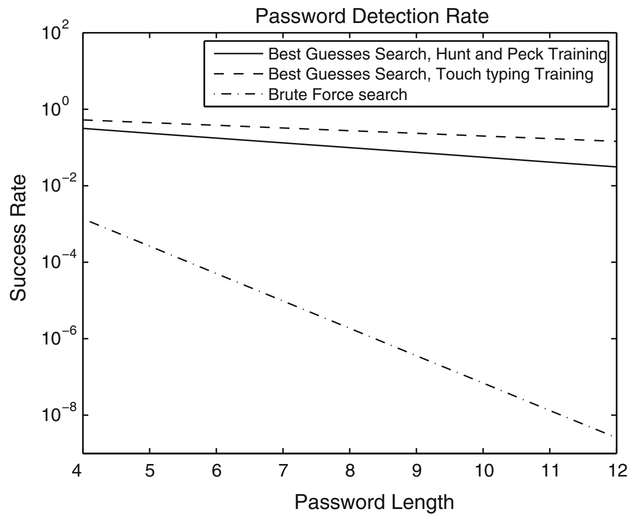


Fig. 10 Best guesses search detection probability

To calculate the password search space size for the Best Guesses Search, the probabilities from Fig. 11 (as per Eq. 5) and the average detection probabilities (as per Eq. 6) are used. The results for passwords of length up to 12 characters are shown in Figs. 9 and 10. These results show that the Best Guesses Search significantly reduces the search space size and improves the detection probability for passwords of different length.

6 Discussion

This research establishes that keyboard acoustic eavesdropping attacks are affected by detection technique, typing style, and type of input data and provides insights about their impact.

6.1 Detection technique

Several techniques were explored for this work, based either on the time signal or on the frequency spectrum. A new technique was further presented which provides improved detection results and is based on both time and frequency data.

The implementation parameters and the performance of the Dynamic Time Warping (DTW) technique were examined. This work shows that the signals do not "stretch" significantly in time which results in the poorer performance of the DTW technique relative to other techniques. In this work, a method of decoding keys based on a distance measure from existing training data was defined and implemented. This method can be used with either time correlation calculations or the Euclidean difference of frequency features. It was observed that the similarities in signals emanated from the same key are detectable both in the frequency and in the time domain. Therefore, a new technique was presented, which combines this information and achieves improved detection results based on both time and frequency data.

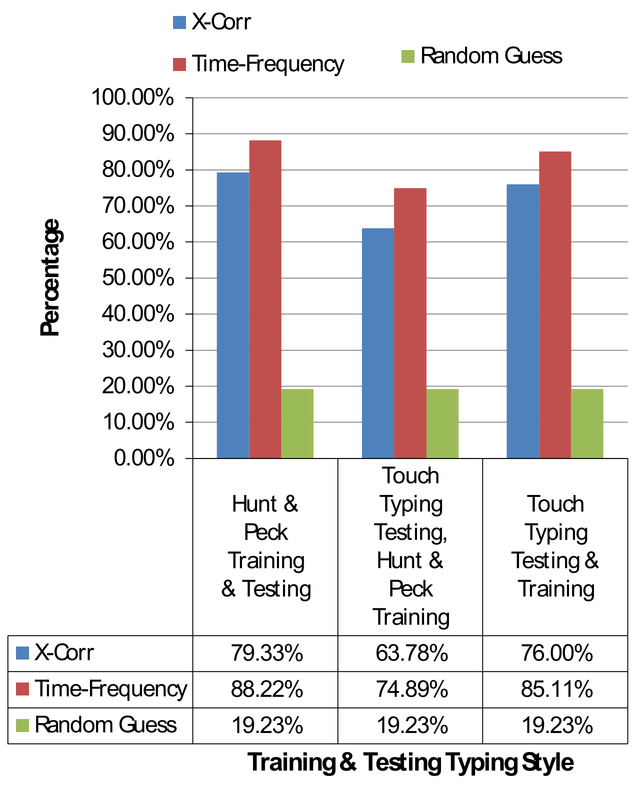
6.2 Typing style

This work demonstrates that while the underlying plate contributes to the key sound, the typing style also contributes to it significantly. The work confirmed that the similarity between the audio sounds belonging to each key is reduced when the typing style changes from hunt and peck typing to touch typing. One of the observations in this work is that while there are still sound differences between some of the keys (which confirms the perception that some keys sound "different"), when examining all the alphabet keys in the keyboard, it becomes hard to distinguish between a single key and all the other alphabet keys.

The experiments in this work showed that the accuracy of detecting a single character on the keyboard is reduced when moving from hunt and peck typing to touch typing (Figs. 6, 11, 12). Therefore, users who employ touch typing are less prone to keyboard acoustic eavesdropping. Since in real-life many users touch type, in practice, keyboard acoustic attacks may not constitute to be as significant a threat as believed to be.

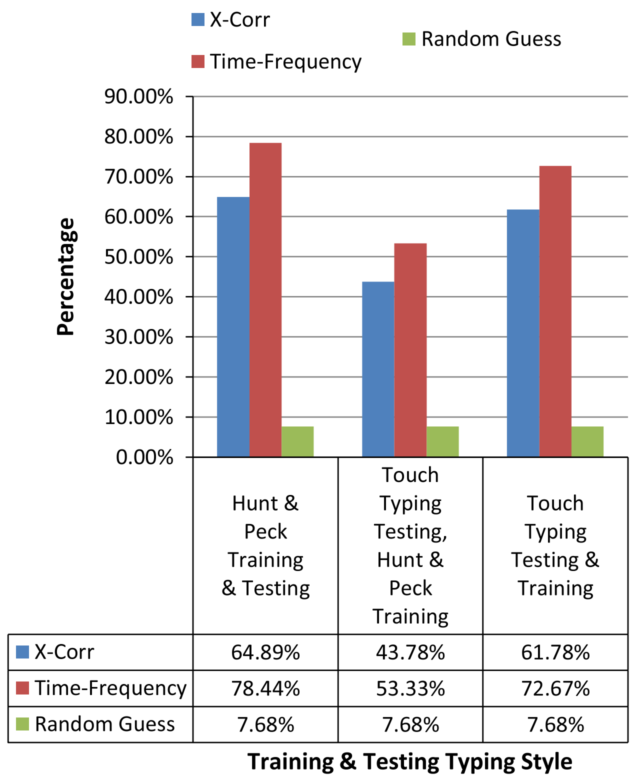
6.3 Type of input data

This research shows that detection of random password poses a significant challenge, since only the (raw) audio signal is available as input to the attack. On the other hand, attacks on English text or weak passwords may achieve better results due to the underlying language model and the dictionary tools, as demonstrated by prior research [6,25,26]. This means that random passwords are less vulnerable to keyboard eavesdropping attacks.



	Hunt & Peck Training & Testing	Touch Typing Testing, Hunt & Peck Training	Touch Typing Testing & Training
🟦 X-Corr	79.33%	63.78%	76.00%
🟥 Time-Frequency	88.22%	74.89%	85.11%
🟩 Random Guess	19.23%	19.23%	19.23%

Fig. 11 Single character detection rates, five character guess



	Hunt & Peck Training & Testing	Touch Typing Testing, Hunt & Peck Training	Touch Typing Testing & Training
🟦 X-Corr	64.89%	43.78%	61.78%
🟥 Time-Frequency	78.44%	53.33%	72.67%
🟩 Random Guess	7.68%	7.68%	7.68%

Fig. 12 Single character detection rates, two character guess

This research concludes that users who employ random passwords are less susceptible to keyboard acoustic attacks than those who employ weak passwords. On the other hand, the attacks presented in this work on random passwords are still orders of magnitude more successful than random guessing or brute-forcing attempts (as depicted in Table 1). For example, with only three trials, for touch-typed passwords, these attacks are better by a factor of about 150,000; with 16,457 trials, they are better by a factor of about 2,000.

7 Other related work

Acoustic emanations were also utilized for eavesdropping on dot matrix printers. In [7], Briol showed that significant information can be extracted about the printed text, using acoustic emanations to distinguish between the letters 'W' and 'J'. In [4], Backes et al. presented an attack which recovers English printed text from the printer audio sounds.

In recent work, Shamir, Tromer, and Genkin [9,18] explore inferring of CPU activities associated with RSA decryption via acoustic emanations to learn the RSA private keys.

In [10], Halevi and Saxena studied acoustic emanations in order to learn key exchange information during the wireless device pairing operation.

Additional methods to extract keyboard input focus on other sources of information (i.e., other than audio). In [5], Balzaroni et al. explored recovering keyboard input based on video of the typing session. In [20], Song et al. showed that timing information of key presses can be used to exploit weaknesses in SSH protocol. In [13], accelerometer data from modern mobile phones (iPhone 4) were used to implement keyboard dictionary attacks.

8 Conclusions and future work

This paper takes a fresh look at the vulnerability of keyboard typing to audio emanations. The work shows that keyboard eavesdropping is affected by a few variables: typing style, input data, and detection technique. It also demonstrates that while the detection performance is reduced for realistic typing styles, keyboard typing still remains vulnerable to eavesdropping attacks.

This paper further provides an objective measure for the performance of key detection. This can be used as a basis for improving future language and dictionary-based attacks (whose success relies on the underlying raw key detection capability) as well as assessing the contribution of the language model to the final detection results.

Overall, the strength of acoustic eavesdropping attacks was found to be limited when using different typing styles and random passwords and may therefore not be as significant a threat as previously believed to be under such realistic and security-sensitive settings. On the other hand, a Best Guesses Search was defined, which reduces by half the entropy of the typed random passwords and therefore considerably speeds-up the exhaustive search.

This work can be extended to also include numbers (e.g., numeric PINs or credit card numbers). Since all the keys are positioned on the keyboard in a similar way and share the same underlying physical plate, the detection behavior is expected to be the same. However, it would be interesting to verify this in future work. Also, looking at the combination of the shift key with other characters is interesting since an overlap is expected between the acoustic emanations of the keys which may make it harder to detect the pressed keys.

Testing laptop keyboard acoustic emanations is another interesting further step. Preliminary tests were conducted by the authors, and it was noted that the press signal is evident in laptop keyboard recordings. However, the release audio signal either had very low volume or was not noticeable at all in the recorded signal. Therefore, laptop keyboard eavesdropping needs to rely only on the key press and is likely to be less successful than traditional keyboard eavesdropping.

Acknowledgments The authors would like to thank Avishai Wool, Yigael Berger, and Doug Tygar for discussions related to prior work on keyboard acoustic emanations. This work was supported in part by the NSF (under Grant No. 0966187). The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of any of the sponsors.

9 Appendix: Algortihms

```
Input: $\mathbf{x}(press)$ and $\mathbf{x}(release)$ key press and release signals of the test sample,
    $\mathbf{y}_i(j, press)$,$\mathbf{y}_i(j, release)$—training data for alphabet letter $i$, $(i = 1...26,\ j = 1...n)$
Output: $A$ index of best matching alphabet key

1. $W \leftarrow$ Output of Algorithm 2
2. $Z \leftarrow$ Output of Algorithm 3
3. $minDTF = \sqrt{2}$
4. for ı = 1..26 do
5.     $DTF_i = \sqrt{DF_i^2 + (1 - C_i)^2} = \sqrt{W^2 + Z^2}$
6.     if $DTF_i < minDTF$
7.         $minDTF = DTF_i$
8.         $A_{min} = i$
9.     end if
10. end for
11. Output: $A = A_{min}$
```

Algorithm 1: Choosing Best Matching Key using Time–Frequency Algorithm

```
Input: $\mathbf{x}(press)$ and $\mathbf{x}(release)$ key press and release signals of the test sample,
    $\mathbf{y}_i(j, press)$,$\mathbf{y}_i(j, release)$—training data for alphabet letter $i$, $(i = 1...26,\ j = 1...n)$
Output: $C_i$ Correlation between the test sample and each alphabet letter $i$

1. Normalize $\mathbf{x}(press)$ and $\mathbf{x}(release)$according to the energy
2. for $j = 1..n, i = 1..26$ do
3. Normalize $\mathbf{y}_i(j, press)$ and $\mathbf{y}_i(j, release)$ according to the energy
4. $C_i(j, press)$
   $= \max(\text{Cross-Corr}(\mathbf{x}(press), \mathbf{y_i}(\mathbf{j}, \mathbf{press}))) =$
   $= 1 - \left[\max_r(\sum_{m=0}^{N-n} \overline{x(press)(m)} \cdot y_i(j, press)(m + r))\right]$
5. $C_i(j, release) =$
   $\max(\text{Cross-Corr}(\mathbf{x}(release), \mathbf{y_i}(\mathbf{j}, \mathbf{release}))) =$
   $= 1 - \left[\max_r(\sum_{m=0}^{N-n} \overline{x(release)(m)} \cdot y_i(j, release)(m + r))\right]$
6. $C_i(j) = 0.5 \cdot (C_i(j, press) + C_i(j, release))$
7. end for
8. Output: $\left(C_i = \frac{1}{n}\sum_{j=1}^{n} C_i(j)\right)_{i=1}^{26}$
```

Algorithm 2: Test Sample-Alphabet Letter Cross-Correlation

```
Input: $\mathbf{x}(press)$ and $\mathbf{x}(release)$ key press and release signals of the test sample,
    $\mathbf{y}_i(j, press)$,$\mathbf{y}_i(j, release)$—training data for alphabet letter $i$, $(i = 1...26,\ j = 1...n)$
Output: $DF_i$ Frequency Distance from sample to each alphabet letter $i$

1. $\mathbf{XFull}(press) = FFT(\mathbf{x}(press))$
2. $\mathbf{X}(press) = \mathbf{XFull}(press)$ coefficients in the [0.4–22.05] kHz range
3. Normalize $\mathbf{X}(press)$ according to the energy
4. for $j = 1..n, i = 1..26$ do
5.     $\mathbf{YFull}_i(j, press) = FFT(\mathbf{y}(j, press))$
6.     $\mathbf{Y}_i(j, press) = \mathbf{YFull}_i(j, press)$ coefficients in the [0.4–22.05] kHz range
7.     Normalize $\mathbf{Y}_i(j, press)$ according to the energy
8.     $DF_i(j, press) = \|\mathbf{X}(press) - \mathbf{Y}_i(j, press)\|$
9.     Repeat procedure for the release signal
10.    $DF_i(j) = 0.5 \cdot (DF_i(j, press) + DF_i(j, release))$
11. end for
12. Output: $\left(DF_i = \frac{1}{n}\sum_{j=1}^{n} DF_i(j)\right)_{i=1}^{26}$
```

Algorithm 3: Test Signal-Alphabet Letter Frequency-Based Distance

References

1. "Keyboard Acoustic Emanations Revisited" presentation. http://cs.unc.edu/fabian/courses/CS600.624/slides/emanations
2. Adams, A., Sasse, M.A.: Users are not the enemy. Commun. ACM **42**(12), 40–46 (1999)
3. Asonov, D., Agrawal, R.: Keyboard acoustic emanations. In: IEEE Symposium on Security and Privacy (2004)
4. Backes, M., Durmuth, M., Gerling, S., Pinkal, M., Sporleder, C.: Acoustic side-channel attacks on printers. In: Usenix Security Symposium (2010)
5. Balzarotti, D., Cova, M., Vigna, G.: ClearShot: Eavesdropping on keyboard input from video. In: Proceedings of the 2008 IEEE Symposium on Security and Privacy (2008)
6. Berger, Y., Wool, A., Yeredor, A.: Dictionary attacks using keyboard acoustic emanations. In: Conference on Computer and Communications Security, SESSION: Attacks and Cryptanalysis, pp. 245–254 (2006)
7. Briol, R.: Emanation: How to keep your data confidential. In: Symposium on Electromagnetic Security for Information Protection, SEPI (1991)
8. Fiona, A.H.Y.: Keyboard acoustic triangulation attack. Final Year Project. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.3156&rep=rep1&type=pdf
9. Genkin, D., Shamir, A., Tromer, E.: RSA key extraction via low-bandwidth acoustic cryptanalysis. In: International Cryptology Conference (CRYPTO) (2014)
10. Halevi, T., Saxena, N.: On pairing constrained wireless devices based on secrecy of auxiliary channels: the case of acoustic eavesdropping. In: ACM Conference on Computer and Communications Security (2010)
11. Inglesant, P., Sasse, M.A.: The true cost of unusable password policies: password use in the wild. In: CHI '10: Proceedings of the 28th International Conference on Human Factors in Computing Systems, pp. 383–392 (2010)
12. Lachlan, R.: Normalization for Dynamic Time Warping. http://luscinia.sourceforge.net/page26/page14/page14.html
13. Marquardt, P., Verma, A., Carter, H., Traynor, P.: iPhone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In: 18th ACM Conference on Computer and Communications Security in Chicago, 2011; proceedings, pp. 551–562. doi:10.1145/2046707.2046771 Key: citeulike:9931496
14. Moore, A.: School of Computer Science, Carnegie Mellon University. Hidden Markov Model. http://www.autonlab.org/tutorials/hmm14
15. Morris, R., Thompson, K.: Password security: a case history. Commun. ACM **22**(11), 594–597 (1979)
16. Rabiner, L., Juang, B.: Fundamentals of Speech Recognition. Prentice-Hall, Upper Saddle River (1993)
17. Rabiner, L., Juang, B.H.: Mel-frequency cepstrum coefficients. Prentice-Hall Signal Processing Series (1993). ISBN:0-13-015157-2
18. Shamir, A., Tromer, E.: Acoustic cryptanalysis: on nosy people and noisy machines. http://people.csail.mit.edu/tromer/acoustic/
19. Shay, R., Komanduri, S., Patrick, K.G., Leon, P.G., Mazurek, M.L., Bauer, L., Christin, N., Cranor, L.F.: Encountering stronger password requirements: user attitudes and behaviors. In: SOUPS '10: Proceedings of the Sixth Symposium on Usable Privacy and Security (2010)
20. Song, D., Wagner, D.,Tian, X.: Timing analysis of keystrokes and timing attacks on ssh. In: Tenth USENIX Security Symposium (2001)
21. Typing. Wikipedia. http://en.wikipedia.org/wiki/Typing
22. Veyrat-Charvillon, N., Grard, B., Renauld, M., Standaert, F.-X.: An optimal key enumeration algorithm and its application to side-channel attacks. In: 19th International Conference, Selected Areas in Cryptography (2012)
23. Wool, A., Berger, Y.: Personal communication on the subject of typing styles used in prior research on keyboard acoustic emanations (2010)
24. Yan, J., Blackwell, A., Anderson, R., Grant, A.: Password memorability and security: empirical results. IEEE Secur. Priv. **2**(5), 25–31 (2004)
25. Zhuang, L., Zhou, F., Tygar, J.D.: Keyboard acoustic emanations revisited. In: Proceedings of the 12th ACM Conference on Computer and Communications Security, pp. 373–382, November (2005)
26. Zhuang, L., Zhou, F., Tygar, J.D.: Keyboard acoustic emanations revisited. ACM Trans. Inf. Syst. Secur. (TISSEC) **13**(1), 3–26 (October 2009)