Eavesdropping over Random Passwords via Keyboard Acoustic Emanations

Tzipora Halevi
Electrical and Computer Engineering
Polytechnic Institute of New York University
thalev01@students.poly.edu

Nitesh Saxena
Computer Science and Engineering
Polytechnic Institute of New York University
nsaxena@poly.edu

ABSTRACT

The sounds resulting from keyboard typing can reveal information about the input data. We revisit such keyboard acoustic emanations for the purpose of eavesdropping over random passwords. In this scenario, dictionary and HMM language models, capitalized in prior work, are not applicable; the attacker can only utilize the raw acoustic information which has been recorded. We investigate several existing signal processing techniques for our purpose, and introduce a novel technique – *time-frequency decoding* – that improves the detection accuracy compared to previous techniques. We also carefully examine the effect of typing style – a crucial variable largely ignored by prior research – on the detection accuracy. Specifically, we compare "hunt and peck typing" with "touch typing", and quantify the impact of changing the typing style on the overall success of the eavesdropping algorithms. Our 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 during the decoding stage reduces the success rate, but using the time-frequency technique, we can still achieve a success rate of around 40% per character.

Our work takes the keyboard acoustic attack one step further, bringing it closer to a full-fledged vulnerability. We explore the limitations of acoustic eavesdropping techniques under realistic and security-sensitive scenarios (different typing styles and random passwords). Our results suggest that while the performance of these attacks degrades under such conditions, it is still possible, utilizing the time-frequency technique, to considerably reduce the exhaustive search complexity of retrieving a random password.

Keywords: *Keyboard acoustic emanations*; *random passwords*; *signal processing*

1. INTRODUCTION

The attacks based on acoustic emanations produced by electronic devices have been a known source of concern. These attacks present a threat to user privacy. Specifically, a few studies examined acoustic emanations of keyboard devices. These studies demonstrate that the seemingly conspicuous sounds resulting from keyboard typing can be used to learn information about the input data. Asonov and Agrawal [2] were the first to extract frequency features from the sound emanations of different keyboard clicks so as to identify the different keys used, utilizing neural networks to classify the resulting keys. This work made an important discovery that since the keyboard has a physical plate beneath the keys, each key produces a different sound depending on its location on the plate (this is very 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. [22, 23] improved upon the attack of [2] by obviating the need for a labeled training recording. Instead, HMM English language-based model [10] was used on a 10-minute typed English text to detect and label the typed keys and decode the text. In this work, a few iterations of feedback-based supervised training using previously classified characters were used to further improve the overall detection accuracy. The authors showed that Mel Frequency Cepstrum Coefficients (MFCC) features [12] yield better classification accuracies compared to the Fast Fourier Transform (FFT) features previously used in [2].

Berger et al. [5] further utilized dictionary attacks to decode 8 letter or longer English words. This attack was implemented utilizing correlation calculations on the recorded signal (in the time domain). For each recorded word, the attack detected a list of best-matching words from an English dictionary. The primary insight of this work is that the keys which are in close physical proximity on the keyboard typically have higher cross-correlation than the keys that are farther from each other.

1.1 Open Research Problems

Our paper takes a fresh look at keyboard acoustic attacks and aims to address some important aspects that prior work left unexplored. First, it investigates 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 [11, 21, 1]. 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 [9, 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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$10.00.

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, we examine the effect of typing style on key detection and eavesdropping ability. Our hypothesis 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 [2]. To our knowledge, ours 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 used the "hunt and peck" or "search and peck" technique [15, 20]. In this technique, as the name suggests, the typist finds and presses each key individually [17]. However, in real-life scenarios, many people employ "touch typing" [17]. Consequently, the second question that we ask 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 drive the research we present in this paper. Our work starts by generating training data and then uses it to classify keys typed by making use of different signal processing techniques. Our focus is on eavesdropping random passwords. We emphasize, however, that in situations where a dictionary or HMM model may be used (e.g., while eavesdropping over English words or text), the techniques we explore are still useful. Our work quantifies the ability of these techniques to provide information about the keys that can be combined with the language models wherever applicable.

1.2 Our Technical Contributions

Our 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 which we outline below.

– **Techniques for Individual Key Detection:** We investigate different signal processing techniques for key detection using acoustic emanations. Specifically, we first consider signal processing techniques which were previously investigated (e.g., the cross-correlation techniques of [5]) and compare them to a well-known technique employed in speech recognition – Dynamic Time Warping [18]. We then introduce a novel technique based on *time-frequency classification*. Utilizing these techniques, we examine the ability to detect the individual key pressed from its recorded signal.

Our implementation of and experimentation with these techniques shows that all the examined techniques achieve significantly better results compared to random character guessing, ranging from 46% to 83% per char (Table 1 provides a summary). We further show that the time-frequency classification technique improves the detection capability and provides the best results compared to all the techniques we studied. We note that these results are applicable to both language text as well as random passwords.

– **Effect of Typing Style on Signal Similarity:** We provide 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. We examine the effect of different typing styles on the similarity between the audio emanations of different presses of the same key, using signal correlation measurements.

Our results indicate that this correlation diminishes significantly when changing the typing style from hunt and peck to touch typing (see Table 2). This leads to our higher level conclusion and key insight that while the location of the key on the physical plate may contribute in part to the audio emanations key similarity, its contribution is limited and the resulting audio emanations are significantly affected by the typing style. This renders the problem of detecting the key typed fundamentally much more challenging in realistic scenarios where people often use touch typing.

– **Random Character (Password) Detection with Different Typing Styles:** We examine the scenario whereby a user inputs a random password, and we determine the probability of detecting each character in the password by eavesdropping using the techniques mentioned previously. In addition, we study different typing styles and the effect they have on detecting the typed random characters. We look at three scenarios: typing each key separately, typing random passwords in a hunt and peck style, and typing the same passwords using a touch typing style. We compare the similarity results corresponding to each typing style and the ability of an eavesdropper to detect which key was typed. We demonstrate that the performance of the key detection techniques reduces significantly when the typing style changes . We further show that our time-frequency classification technique produces better results for the different typing styles (see Table 3).

Our research further shows that using data typed with touch typing style during the training phase improves the detection probability of data typed with the same typing style (but is still lower than the data typed as well as trained using hunt and peck style) (Table 3). This shows that the typing technique affects the resulting audio signal, and that the algorithms which use the data typed with the same typing style – during both training and decoding phases – provide significantly improved detection capability. This insight is again applicable to both language text as well as random passwords.

Finally, we suggest two methods for performing exhaustive search which significantly reduce the password search space while considerably improving the detection capability over random search. Specifically, we reduce the search space size to only 5 letters per character (from 26 letters per character). This reduces the search size by a factor roughly 2, from 4.7 bits per character to 2.3 bits per character, while still improving significantly the detection rate (compared to brute-force attack).

– **Broader Implications of Our Work:** We note that in real-life, users employ different typing styles. Specifically, touch typing is one of the more popular typing techniques used. We examine the effect of different typing techniques on the key detection in cases of random passwords where, as previously mentioned, a dictionary attack or language-based model is not useful. 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. We also note that our attack gives an objective indication as to the amount of data that eavesdropping can provide about the keys. While utilizing a language-based model can be very helpful to an eavesdropper, its success is dependent on choosing an appropriate model for the typed data context.

Paper organization: The remainder of this paper is organized as follows. We start by defining our threat model in Section 2. We continue in Section 3 by describing the different techniques used to detect pressed key and the performance of these techniques. We then describe, in Section 4, our experiments for testing the effect of different typing styles for eavesdropping over random passwords,

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

followed by the performance of our password detection techniques in Section 5. Next, we discuss and interpret our results in Section 6. Finally, in Section 7, we review some other work related to acoustic emanations and password attacks.

2. THREAT MODEL

Our attack model is very similar to the one considered by prior research on keyboard acoustic emanations [2, 22, 23, 5]. Basically, we assume that the adversary has 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.

Our model, however, differs from prior research since data of interest to the adversary is a random password typed by the user, and not English (or text in other languages) or weak passwords susceptible to dictionary attacks. This results in a more challenging set-up for eavesdropping due to the fact that the adversary can not use a language-based model or dictionary to decode data. In our experiments, we consider random passwords consisting of lowercase English alphabets. We assume 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]

Our attack examines 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 our system. We emulate this scenario by using both training and testing data typed with the same typing style (this is done in two typing styles, as mentioned in section 4).

Another possibility, which we employ in our attack, 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. To emulate this possibility, our training data is captured in a "mechanical" style (discussed in Section 4.1) which maximizes the effect of the underlying keyboard plate on the recorded sound. This 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). In this case, the attacker will use the "mechanical" training data to decode the tested text.

We emphasize that since an HMM language-based model or dictionary can not be used for the attack, and since passwords may be as short as 6 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, we assume that the attacker has access to the device or a service that needs authentication (e.g., a personal desktop or a web-site) for a limited amount of time. The attacker is usually allowed to make a certain number of password trials to determine the correct password. We therefore suggest a method of password exhaustive search that reduces significantly the overall search space while increasing the probability of correct password detection. We also provide the success probabilities of finding the correct password when the attacker is only allowed up to three trials, 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.

Attack Set-Up and Tools: Throughout our experiments, we used a standard Lenovo keyboard (model JME7053 English) for our typing needs and for producing the acoustic data as an input to our algorithms. We used a standard inexpensive PC microphone for recording the keyboard acoustic signals and a Thinkpad X60 laptop computer for our development and evaluation work. To record the samples, we used the RecordPad software (v.3.03). For the signal processing computations, we used the Matlab software. These off-the-shelf equipment and tools were used, similar to the prior research [22, 23, 5], so as to maximize the overall impact (feasibility) of the underlying attacks, which can possibly be perpetrated by an unsophisticated adversary. The random password characters were generated using the Matlab's "rand" command that yields uniformly distributed pseudo-random numbers. (The full password can be generated with the "char('a' + ceil(rand(1,6) * 26) - 1)" Matlab script).

3. TECHNIQUES FOR INDIVIDUAL KEY DETECTION

To develop our attack algorithms, we started by exploring techniques for the detection of individual keys/characters pressed on the keyboard.

To this end, we examine the use of Dynamic Time Warping (DTW) technique [18] – which has been widely used in speech recognition – for key detection. We then compare the performance of the DTW technique to that of previously used techniques for key recognition. Specifically, we utilize the cross-correlation based technique employed in [5] and frequency-based feature extraction using neural networks employed in [2] as well as suggest a method of using frequency-based distance (similar to the technique used in [5]) and analyze their suitability for detecting the characters pressed. We next introduce a novel time-frequency based classification technique for individual key detection. In this method, we combine both the cross-correlation and the frequency spectrum features to choose the best matching key. We measure the performance of this method for individual key detection and eventually for strong password detection (as will be discussed later in Section 5).

3.1 Determining Key Press Signal

Keyboard acoustic signals have two distinct regions: push (also referred to as press) and release (Figure 1), as demonstrated in [2]. 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, our experiments detect that depending on the force sustained while pressing the key, both the push and the release have between 1 to 3 distinct peaks. We first examined using short regions of the signal to measure the correlation between signals generated by the same key. We compared between two cases: choosing the first peak in the push region versus using the most pronounced one. We found that the most pronounced peak gave the best result. We then experimented with using larger regions versus short regions and found that for regions of 50 ms (which

[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.

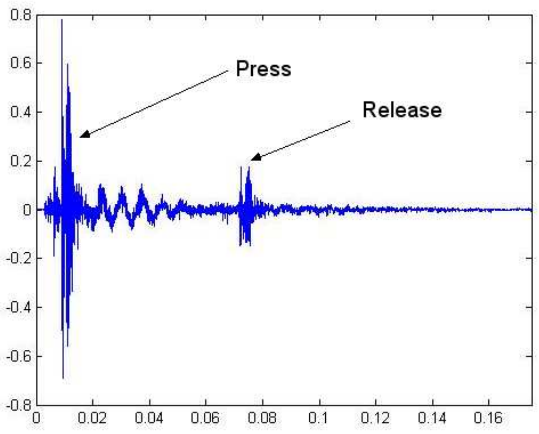

Figure 1: Acoustic Signal of a Single Key

started from the beginning of the first press and release regions), the results obtained were the best. The reason seems to be that in this case, the region includes all of the signal data produced by the key press.

Detecting Key Press Regions: We record our signals with a sampling frequency of 44.1 kHz. To detect the beginning of each press, we calculate the FFT (Fast Fourier Transform) coefficients of the signal using a window size of 440 samples. We then sum-up the FFT coefficients in the range of 0.4-22 kHz and use a threshold to detect the beginning of each keypress (Figure 4 and Figure 5). To detect the key release region, we examine the area following the push region (first 50 ms section of the signal). Since the release is less pronounced, we calculate the FFT coefficients for the rest of the region using a smaller window size of 88 samples. We then sum-up again the FFT coefficients and use a threshold to determine the beginning of the release region. As stated in Section 2, we utilized the Matlab software to implement all our signal processing calculations. For calculating the FFT coefficients for each signal, we used the Matlab "specgram" command.

3.2 Technique 1: Dynamic Time Warping

DTW is an algorithm which measures similarities between sequences. We examined the capability of the algorithm to detect the similarities between recordings of different instances of the same key (while distinguishing instances of different keys). We used the simple distance measure between each two elements in the signal vectors to calculate the difference between them. We tested the algorithm on both the press and release regions of the signal. We implemented the main program in Matlab, but used a C source code for a Matlab executable (MEX) for the DTW function implementation, which resulted in a faster overall running time. We experimented with different signal length for the press and release, and found that for the DTW technique, longer regions (50 ms) produce better results (compared with shorter regions).

Our experiments showed that the audio key signals have varying amplitude depending on the strength with which the keys were pressed. Since correct normalization is essential for DTW, we tried using different methods of normalization (as well as no normalization). We tried normalizing according to the amplitude, mutual joint distribution [14], and energy. We found that normalizing according to the energy (normalized to 1). provided superior results to the other methods and is inline with the normalization used for the cross-correlation technique employed in [5]. We also examined the results for using DTW only on the push period, release period and on a mean of both. We found that using the mean of the algorithm results calculated for the push and release periods gave the best results.

Letter Data Set We created a training dataset for each letter to be used for decoding. Each Letter Data Set is made up of n samples that are typed for the corresponding alphabet letter.

The DTW technique produces a distance measure between each two signals. To match each test key with an alphabet key, we examine all the instances in the training data and compare between two possibilities. In the first case, we pick the closest training instance (the one with the smallest distance to the test character) in the training data and match its corresponding alphabet key. In the second case, we calculate the average distance from the test character for each alphabet key. This is done by getting the mean of the similarity measurement for the test character and the Letter Data Set of each alphabet key. We then pick the best match as the alphabet key which has the smallest average distance to the test key. We found that using the mean value for each training alphabet key produced better results, since it minimizes the noise contribution of each single test instance.

3.3 Technique 2: Cross-Correlation

Cross-Correlation (denoted X-Corr) is a commonly used signal processing technique for measuring similarity between signals. As done in [5], we performed the cross-correlation between the recorded signals. We first normalized the signals according to the energy level. For each pair of signals, we calculated the cross-correlation between their press regions and took the maximum value. We repeated the process for the release regions. We then took the average of the two values and used it as our similarity measurement.

To match each test instance to an alphabet letter, similar to the DTW technique, we compared between two cases. In the first case, we chose the key which belongs to the closest instance in the training data (the one with the highest correlation). In the second case, we calculate the average similarity measurement for each alphabet key. This is done by taking the Letter Data Set for each alphabet key and calculating the average of their cross-correlation measurement, receiving one similarity measurement for each alphabet key. Then, we chose as the matching alphabet letter the one with the highest similarity measurement. We found that for the cross-correlation technique, choosing the second option produced better results as well (similar to the DTW technique).

Therefore, to determine the alphabet letter belonging to our test character, we chose – as the best match – the key which gives the highest cross-correlation to our training signals.

We also compared using the mean between the press and release cross-correlations to using only the press or release cross-correlation. Our tests confirmed that the mean between the press and release values produced the best performance (similar to the results in [5]).

Another variation we tried was matching the peaks of both signals and calculating the correlation, similarly to the technique described in [7]. Our experiments showed that this produces worse results than taking the maximum correlation (contrary to what was found in [7]). We therefore continued all our experiments using the max correlation.

3.4 Technique 3: Frequency-based Distance Measure

The Frequency-domain Features-based Distance Measure (denoted Freq-Dist) technique is similar to the one described in [5].

However, instead of using only a small part (typically 2-3 ms) of the press and release signals, we take the full 50 ms region for both the press and release region. We then produce the frequency spectrum for both regions. We examined using different bands of the frequencies but found that the best results were achieved when using the coefficients in the 0.4 - 22 kHz. For each of the signals, we normalize the coefficients to one.

We compute the frequency-based distance between each two signals by calculating the Euclidean difference between those features. We repeated this for both the press and release parts of the signal and calculate the mean of both to get a single distance measure.

We use the distance measure between the signals to calculate the average distance between the test character and the Letter Data Set of each alphabet letter (as described in Section 3.3). We chose as the best match the alphabet letter which gives the smallest distance to the test character.

Our experiments show that this technique produces poorer results than the cross-correlation technique, but produces significantly better results than the DTW technique (please refer to the next subsection for our performance results). We conclude that different instances of the same key produce similar spectrum when examining the full press and release signal.

3.5 Technique 4: Frequency-Domain Features with Neural Networks

We implemented the frequency-domain features based technique, described in [2], and tested it on our data. To detect the most active 3 ms window corresponding to the press and release regions, we used the algorithm described in Section 3.1 with a window size of 3 ms (132 samples). We calculated the signal spectrum and summarized the FFT coefficients over the 0.4-22 kHz, using a threshold to detect when the peak press and peak release began.

After creating the frequency-domain features, we used the Matlab Feed-Forward Neural Network to classify the keys. However, we were not able to reproduce the results described in [2]. Our experiments indicated that this technique was worse than either DTW, the correlation techniques or the frequency-based distance technique (which we will discuss in the following subsection).

We note that previous research [22, 23] also could not achieve results similar to the original ones described. The difference in the findings could be due to different reasons. (We did not have access to the original data used for those tests.) The keyboard used for our tests is different from the one used in the experiments of [2] (which may have produced acoustic emanations with higher volume or more pronounced characteristics). Another difference could result from the fact that we are using the Matlab neural network while the original research used Java neural network. Also, the authors of [2] did not specify how to choose the press and release regions. We chose an automatic method 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.

3.6 Performance

The techniques were evaluated on the training data, which was taken with the hunt and peck style and is described in Section 4.1. This style minimizes the noise due to the fingers touching other keys and the overlap of other key presses.

While conducting our experiments to determine performance of the techniques, due to the relatively high computation requirements for the DTW algorithm, we used only four instances per alphabet letter as our training data. We then tested the ability of the techniques to pick the correct key pressed out of 26 alphabet letters.

Table 1: Single Character Detection

Method	Detection Rate
DTW	46.15%
X-Corr	73.08%
Freq-Dist	63.46%
Time-Frq	82.69%

Overall, we found that the cross-correlation technique gave the best results, with a single key detection rate of 73%. The frequency-based distance measure produced a lower detection rate of 63%.

For the DTW algorithm, the detection rate was 46%. We note that if we were to pick the key value randomly, our chances of picking the correct key would be less then 4%. Thus, both DTW and cross-correlation significantly raise the ability to chose correctly the matching key. The detection results for the training data, using four instances per key, can be found in Table 1.

For better visualization, we show, in Figure 6 and Figure 7, the decoding results for the different alphabet letters (values which lie on the line Y=X are successful identifications. Each letter was typed four times and is represented by its index ($'a' = 1$ to $'z' = 26$). Each row shows the decoding of the four instances of the corresponding test alphabet letter (i.e., which training letters were found to be the best matches for each of the four instances tested). The brighter the rectangle, the more instances were assigned to the same alphabet letter. Values which lie on the line $Y = X$ indicate successful identifications.

DTW is used to measure similarity between two sequences which vary in time or the speed in which they occur. Since the audio key signals are affected by the typing technique, we examine the possibility this causes warping in the resulting press and release signals, and different instances of the same key may include delays which affect the output signals. However, our experiments show that the DTW technique gives lower detection results compared to the correlation algorithm. This indicates that the key press and release do not vary much in time, even when the typing technique changes. As a result, using this technique causes reduction in the differences between presses of different keys in the keyboard, making it harder to distinguish between them and therefore raising the error rate when choosing the "best match" for each typed key.

3.7 The New Technique: Time-Frequency Classification

In the time-frequency classification method (denoted Time-Frq), we combine both the correlation calculation and the frequency-based calculations to choose the best-matching letter for each training letter. We start by calculating the frequency-domain features-based distance measure for each instance, as described in section 3.4.

For each of our test samples, we took 4 instances of each typed key from our training data. We calculated the frequency distance between each test character and the Letter Data Set of each alphabet key, for both the key press and the key release. We then calculated the mean between the press and the release value and obtained a final frequency distance value F for each key combination.

Similarly, we calculated the cross-correlation for all the test character with each alphabet key and obtained the cross-correlation C value for each key combination.

We note that when calculating the frequency difference cross-correlation for the signal with itself, we get $F = 0$ (which is the minimum possible value for any two signals). Similarly, we get that the cross-correlation is $C = 1$ (which is the maximum possible cross-correlation value for any two signals). Since we are looking to combine both elements, we calculate $MC = 1 - C$. At this point, both F and MC are ascending with a minimum of 0 for the distance of a signal with itself.

At this point, we examined a few methods of combining both matrices. We tried picking the minimum of each value $(\min(MC, F))$ and the average of the two values (MC, F). We also looked at (F, MC) as a point on a 2-D space and calculate the Euclidean distance from zero. We found that the best results were achieved using the last method. We therefore use the Euclidean distance as our distance measure for classifying each key (denoted as TF).

To this end, we examine each test sample and calculate the TF distance measure to all of the alphabet letters in the training data. We then chose as the best match the alphabet letter which yields the lowest distance measure (corresponding to the point closest to zero).

Using the time-frequency classification technique, we get an increased probability of 83% for the training data (please refer to Table 1 for comparison with other techniques). This technique combines the information in both the time (cross-correlation) and the frequency spectrum. We observe that even though the cross-correlation provides better information in most cases, in some of the cases, the time-based signal changes but most of its frequency characteristics are still evident. The time-frequency classification technique incorporates this information into the key classification process. We therefore conclude that both the frequency and the time data can be used together to produce better results. In Figure 8, we provide a visualization for the decoding results corresponding to different alphabet letters in case of time-frequency classification.

4. RANDOM PASSWORDS AND TYPING STYLES

To recall, in this paper, our goal is to determine the advantage that an attacker can have by using key detection techniques (studied in the previous section) to eavesdrop over random passwords. We further aim to examine the effects of the typing style, i.e., hunt and peck vs. touch typing, on the detection ability.

To this end, we first create the training data. The training data produces the "best" sounds that the audio emanations can provide - i.e., using the hunt and peck and always hitting the keys from a vertical position. This maximizes our ability to capture the sounds emanated from the physical plate underneath the keys and minimizes the effect of the interaction between the keys and the fingers. We recall that this typing style has been employed in previous research and is expected to produce the best similarity between the audio emanations of the keys.

4.1 Straw Man Approach: Typing Each Key Separately

Our first scenario involves typing each letter multiple times always using the same finger. In this scenario, each letter is typed a few times continuously before moving to the next letter and a few seconds are allowed before typing the next letter (similar to the technique used in [2]). 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 enabled 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 any audio signal noise or overlap sounds during the key press and therefore maximized the contribution of the keys hitting the underlying keyboard plate. Since this plate acts like a "drum", it produces the emanated audio sound ([2]). Since each key is positioned differently on the keyboard plate, different keys will produce a different sound.

In addition, this technique maximizes the contribution of the underlying plate on the audio emanations (relatively to other factors) while minimizing the typing style contribution. This technique can therefore be used to train the system by an attacker (not the original typist) who is trying to get information about the audio emanations of the keyboard which are due to the physical structure of the keyboard and the differences between the location of its keys.

We used the above technique to take ten signal recordings for each key of the alphabet letters as our training data. We refer to this data in the rest of the paper as "Train Hunt and Peck data".

4.2 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, we chose to generate random passwords of 6-character each (since the characters are chosen randomly, the data could be divided into any password size). Since 6-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. We generated a total of 25 different such random passwords, and each password was typed 3 times consecutively. We refer to this data as the "Test Hunt and Peck data" in the rest of the paper.

4.3 Touch Typing

In the third scenario, we type the same password list – 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). We recall that this typing technique is very popular among users. However, this typing style does affect the acoustic emanations of the key as the key is hit from different angle, 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). In addition, since both hands touch the keyboard at the same time, there are overlapping sounds from the other fingers as they release the previous keys and are re-positioned on the original middle keys. We refer to this data as the "Test Touch Typing data".

4.4 Effects of Typing Style on Signal Correlation

To measure the effect of typing style on the detection of typed random password, we examine the maximum correlation between instances of the keys in the test data with the Letter Data Set of the same keys in the training data, termed as *matching keys*. We then compare it to the correlation with the Letter Data Set of the rest of the keys in the training data, termed as *non-matching keys*. We note that for each signal, the correlation with itself is normalized to 1. We therefore expect the max correlation with any other signal to be between 0 and 1.

Our training data included 10 training samples using the straw man typing approach (described in Section 4.1), when each key was typed separately multiple times. Recall that this causes the key to be hit from a vertical angle each time and reduces the variability within the signal.

Straw Man Typing: We started by using the aforementioned data as test data itself. For each sample, we calculate the maximum corre-

Table 2: Probability of Keys Matching the Training Data with Typing Style Variation

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

lation with that sample itself and with each of the other instances taken with the same key. We then calculate the mean of these values. We did this for both the press and release part of the signal. We mark these values as $\mathsf{PcorrMatchPrs}(i)$ and $\mathsf{PcorrMatchRls}(i)$ for each sample i of the data.

We also examined the correlation between samples taken with each key and the Letter Data Sets of the rest of the keys. Specifically, for each key signal, we calculated the maximum correlation to each instance in the training data. We then calculate the average correlation of the sample to the Letter Data Set of each alphabet letter.

For each tested sample, we take the highest value of the 25 values we received, which shows the correlation to the most likely key to be chosen as a match to the original sample. We mark this value as $\mathsf{PcorrNonMatchPress}(i)$ and $\mathsf{PcorrNonMatchRls}(i)$ for each sample i of the data. At this point, we compare the correlations of the press and release samples. If the sample has highest correlation to the other instances of the same letter, i.e.,

$$\mathsf{PcorrMatchPrs}(i) > \mathsf{PcorrNonMatchPrs}(i), \qquad (1)$$

we mark the press part of the sample as a Match correlation. We do the same for the release part of each signal. If, on the other hand, the instance has higher correlation to any of the other letters in the training data, we mark it as Non-Matching. Since, in this case, the key press (or release) is more similar to instances of another letter in the training data and therefore will be decoded as the non-matching letter based on this press (or release) sample. We calculate the Match probability 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. For the training Straw Man Typing, we found that 56% of the press signals and 67% of the release signals best matched their corresponding typed letter, which shows a relatively high correlation between the presses and releases of the different samples belonging to the same letter.

Hunt and Peck Typing: We next performed the above analysis for the passwords data typed in hunt and peck style. We found that for this case, the Match probability was reduced to 28% for the press samples and 43% for the release samples.

We therefore observe that the percentage of signals that are best correlated to the training data belonging to the matching letter is significantly reduced for both the press and release samples.

We see that when the typing style changed slightly (since the keys are not hit from the top anymore in a monotonic fashion but rather may hit the keys from different angles, depending on the previous letter in the password), the correlation between instances of the same key reduces compared to the correlation to instances of the other keys. Therefore, it is more likely to choose the wrong key as the best matching key to the new sample.

Touch Typing: We further repeated the analysis for the data taken with the touch typing style. We calculated the correlation between these samples to the training data. In this case, we found that 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 training part of the signal.

A summary of the results of our overall analysis is presented in Table 2. In conclusion, we observe that the maximum correlation is stronger between instances of the same key taken with the same typing style but reduces when the typing style changes. On the other hand, the correlation to instances taken with other key increases which makes it hard to detect correctly the pressed key. This confirms our hypothesis that typing style has a strong effect on the similarity of audio signals taken with the same key and the ability to distinguish them from other keys in the keyboard.

We note that this insight applies to both random passwords as well as language text.

5. PERFORMANCE OF PASSWORD DETECTION TECHNIQUES

Out of the five techniques explored in Section 3, we found that the cross-correlation (X-Corr) and time-frequency classification (Time-Frq) techniques yielded higher accuracies. In this section, we investigate the advantage that an attacker can get by using these two techniques to eavesdrop over random passwords.

We examine the performance of these techniques when the typing styles changes. We compare the detection rates – using the training data (specified in Section 4.1) – for random passwords typed with both the hunt and peck and the touch typing styles.

We start by examining the key detection rate for each of the data groups. We utilize as training data ten instances of each alphabet key (as opposed to four instances used in Section 4.1). This improves the detection ability as it helps in reducing the effect of noise in each instance.

We calculate the similarity measure – max correlation for cross-correlation and TF distance for time-frequency classification – between each tested instance and each of the ten instances of the training data. We then calculate the average of the similarity measurement between each tested instance and all instances belonging to each alphabet letter in the training data. For the cross-correlation technique, we chose as the best matching letter the one with the highest correlation. For the time-frequency classification technique, we chose the alphabet letter with the lowest TF distance measure.

5.1 Training Data, Hunt and Peck style

To measure the performance for our training data, we use the same method as discussed in Section 3.3. Since we now raise the number of instances for each alphabet key to ten, this averages the noise per instance and improves the detection performance.

As a result, we found that the cross-correlation statistics calculated using this technique resulted in a 83% accuracy rate per key (up from 73% when using only four instances per alphabet key). We conclude that when the typing is repetitive, the probability of detecting each key using cross-correlation is relatively high and the underlying physical characteristics of the keyboard has strong effect on the acoustic emanations and the ability to eavesdrop on the recorded characters. When using the time-frequency based classification, we found that the results were further improved to 89% .

5.2 Test Data, Hunt and Peck Style

For calculating the detection rate for the password (test) data, we start by calculating the similarity measure (max correlation and time-frequency distance) for each character in the password. To do this, we begin by calculating the similarity to all the instances

Table 3: Single Character Detection Rates, best character guess

Training →	Hunt & Peck		Touch Typing
Testing Stage →	Hunt & Peck	Touch Typing	Touch Typing
X-Corr	53.78%	33.78%	49.33%
Time-Frq	64.67%	40.67%	58.89%
Random Guess	3.84%		

Table 4: Single Character Detection Rates, 5-character guess

Training →	Hunt & Peck		Touch Typing
Testing Stage →	Hunt & Peck	Touch Typing	Touch Typing
X-Corr	79.33%	63.78%	76.00%
Time-Frq	88.22%	74.89%	85.11%
Random Guess	19.23%		

in the training data. We then calculate the mean of the similarity measure for each alphabet letter in the training data (by averaging the values received for all the instances of each letter). For the cross-correlation technique, we chose the matching letter as the one with the highest cross correlation for each test instance.

For the password data typed with the hunt and peck typing style, we find that the cross-correlation performance is reduced to a 54% accuracy rate per key. We see that the typing style causes a reduction of the detection accuracy compared to typing the same key continuously. We conclude that the angle at which the finger hits the key affects the acoustic signal emanated by the key and the ability to detect correctly the key by comparing it to the training data.

When employing the time-frequency classification technique, we chose as the best matching alphabet letter, for each test instance, the one which produces the smallest TF distance. We found that our results were improved to a detection rate of 65% per character in this case.

5.3 Test Data, Touch Typing Style

We repeated the testing process for the passwords typed using the touch typing style. We find that for this scenario, using the cross-correlation technique for key detection, the accuracy rate is reduced to 34%. When using the time-frequency based classification, we observe that the rate of detection per correct character has increased to 41%.

5.4 Best Guesses Search

In order to raise our detection rate, we decided to create a list of additional keys to be checked against our recorded password. We implemented this by creating a list of keys having the highest max correlation, for the cross-correlation technique, to each recorded password character. For the time-frequency classification technique, we created a list of keys having the lowest TF distance from the test character.

For each key, we tried matching both the best matching letter and the next few highest matching alphabet letters. When examining the ordered list of highest matching alphabet letters, we saw that the probability of the key matching each of the letters reduces significantly after the fifth letters.

We therefore implement a **"Best Guesses Search"** – in which for each typed character, we create a list of the 5 best matching keys. We then determine the probability of a correct detection for the five keys. Using the correlation-based technique, we found that the probability of each character to be in the list of the top five keys was 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, we found that the probability of each character to be in the list of the top 5 keys was increased to 88%. For the touch typing data, the rate was increased to 75%. All of our results, corresponding to the single best key, and the first five keys produced by our algorithms, are summarized in Tables 3 and 4, respectively. The results corresponding to the two best guesses are depicted in Table 6 (moved to appendix due to space constraints).

5.5 Training, Touch Typing Style

We now examine the case where the training is also performed using continuous typed characters. 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, we first type each letter continuously and record the audio signal, using touch typing. We then use the typed signals to create the training data. We then use both the cross-correlation and the time-frequency method to decode the password data based on the recorded training data.

Our tests show that as expected, the password decoding has significantly improved in this case (compared to training with hunt and peck style data). We also found that, in this case too, the time-frequency method produced better results than cross-correlation. We summarize the results in Table 3.

5.6 Password Decoding

Next, we look at 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). While a brute-force attack on the entire password space would take $26^n \simeq 2^{4.7\times n}$ trials, we introduce the Best Guesses Search, which includes a lower number of tests. This reduces significantly the computing complexity and speeds-up the attack. Therefore, it can be used by an attacker who has access for a limited amount of time to a device (or a service) that requires password authentication. We also provide the success probabilities of finding the correct password when the attacker is only allowed up to three trials, as it is a common practice among many online services to lock out the owner's account after three failed attempts.

We start by analyzing the Best Guesses Search. For this algorithm, we choose for each character the five keys which are the closest (the most similar) from all the keys in the training data. We therefore reduce the number of tests to:

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

This yields a probability of detecting the full password with the Best Guesses Search as:

$$Pr_{PasswordDetection}(n) = (\mathsf{Pchar}_5)^n \tag{3}$$

Here, Pchar_5 denotes the probability that each char matches one of the five best guesses.

Overall, this means that our attack can cut down the entropy of the password search space by a factor of about 2 (from 4.7 to 2.3).

We further compare the accuracies of the cross-correlation and time-frequency classification techniques for detecting $n-character$ passwords for the Best Guesses Search as well as when performing a small number of trials. We check, as a test example, the Best Guesses Search detection probability for 6-character passwords (which

Table 5: 6-Character Password Detection Rates

Method →	Exhaustive Search						Brute Force
	Cross-Correlation			**Time-Frequency**			
Training Stage →	Hunt and Peck		Touch Typing	Hunt and Peck		Touch Typing	
Testing Stage →	Hunt and Peck	Touch Typing	Touch Typing	Hunt and Peck	Touch Typing	Touch Typing	
No. of Trials ↓							
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%

includes $5^6 = 15625$ trials) and verify that it indeed matches our calculations using Equation 3. We start by calculating this probability using the cross-correlation techniques. When using the hunt and peck training on the password typed with the hunt and peck technique, we managed to detect 24% of the passwords. For the passwords typed with the touch typing technique (using the hunt and peck training), on the other hand, the detection rates went down significantly – to only 5.3%. We emphasize, however, that our results are still considerably better than a brute force attack which would produce on average 0.005% success rate for the size of our search space (which includes about 2^{14} password tests). When examining the average detection rate for the case where touch training is performed, we find that the average password rate is raised to 22.7%.

We then compare our results to the one achieved with the time-frequency classification. For the hunt and peck typing, we obtain a detection rate of 42.7%. For the touch typing techniques passwords (using the hunt and peck training data), the results improve using the time-frequency technique and the detection rate reached 21.3%. For the case where we use touch training on this data, we find that the average password rate for the touch-typing passwords goes up to 34.7%.

All of the detection rates are summarized in Table 5. 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 Tables 3, 4 and 6.

Finally, we use the probabilities from Table 4 to calculate the password search space size (as per Equation 2) and the average detection probabilities (as per Equation 3) for the Best Guesses Search. We show the results for passwords of different lengths up to 12 characters in Figures 2 and 3. We therefore see that the Best Guesses Search significantly reduces the search space size and improves the detection probability for passwords of different length.

6. SUMMARY AND IMPLICATIONS OF RESULTS

Our research establishes that keyboard acoustic eavesdropping attacks are affected by three variables, namely, detection technique, typing style, and type of input data. Below we outline some of the insights that our research provides vis-a-vis these variables.

6.1 Detection Technique

We explored several techniques, based both on the time signal and frequency spectrum. We further present a new technique which gives improves detection results and is based on both time and frequency data. Our work also examines other potential time-based detection techniques and the usage of Dynamic Time Warping technique for key detection. Our work shows that the signals do not "stretch" significantly in time which results in the poorer performance of Dynamic Time Warping technique compared to signal time correlation (Table 1). We conclude that while the audio signal changes from click to click, the changes affect mainly the audio signal amplitude. We further observe that the similarities in signals emanated from the same key are detectable both in the frequency and in the time domain. Therefore, combining this information leads to improved detection results, as is the case with our time-frequency classification.

6.2 Typing Style

Our work demonstrates that typing style significantly affects the emanated keyboard sound. We further conclude that while the underlying plate contributes to the key sound, the typing style also contributes to it significantly. We 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 our observations is that while there are still sound differences between some of the keys, when examining all the alphabet keys in the keyboards, it becomes hard to distinguish between a single key and the rest of the keys. While it may be easier to distinguish the key from some of the keys (which confirms our perception that some keys sound "different"), distinguishing it from the rest of all the alphabet keys is challenging and some of the audio emanations corresponding to other keys may become indistinguishable from the target key.

We found, from our experiments, that the accuracy of detecting a single character on the keyboard is reduced when moving from hunt and peck typing to touch typing (Tables 3, 4 and 6). An implication of our result therefore is that users who employ touch typing are less prone to keyboard acoustic eavesdropping. Since in real-life many users touch type, this suggests that, in practice, keyboard acoustic attacks may not constitute to be as significant a threat as believed to be.

6.3 Type of Input Data

Our 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 [22, 23, 5]. Such attacks may achieve better results because the raw acoustic information can be clubbed together with an optimized context-based language models or a dictionary. This means that random passwords are less vulnerable to keyboard eavesdropping attacks.

We can conclude that users who employ random passwords are less susceptible to keyboard acoustic attacks than those who employ weak passwords. On the other hand, our attacks on random passwords are still orders of magnitude more successful than ran-

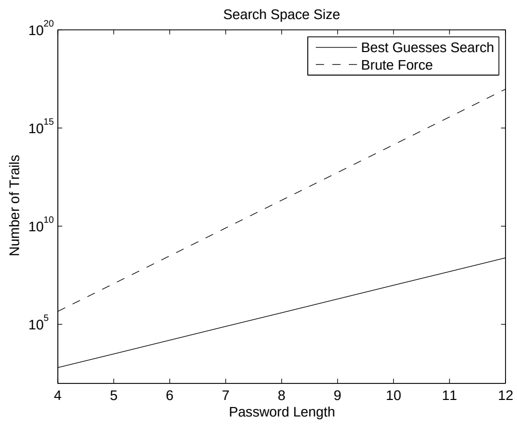

Figure 2: Best Guesses Search space size

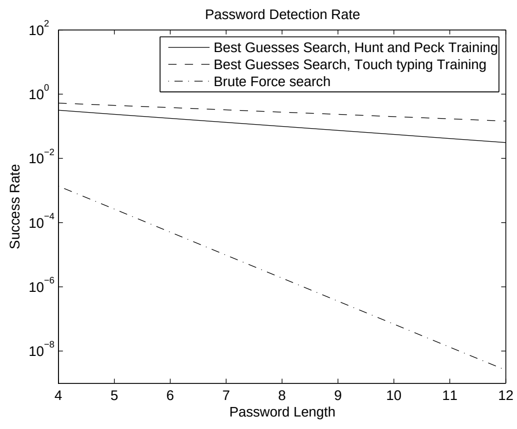

Figure 3: Best Guesses Search detection probability

dom guessing or brute-forcing attempts (as depicted in Table 5). For example, with only 3 trials, for touch typed passwords, our 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 [6], Briol showed that significant information can be extracted about the printed text, using acoustic emanations to distinguish between the letters 'W' and 'J'. In [3], Backes et al. presented an attack which recovers English printed text from the printer audio sounds. This attack is word-based and starts by training the system with a list of dictionary words. Then, an HMM language-based model is utilized to detect the typed words from the dictionary word list.

In a proof-of-concept work published on the web [13], Shamir and Tromer explore inferring of CPU activities (e.g., patterns of CPU operations and memory access) via acoustic emanations. In particular, they investigate how acoustic emanations associated with RSA decryption and signing operations produce unique signatures per RSA private key, and how they can be used to learn the keys.

In [8], Halevi and Saxena studied acoustic emanations in order to learn key exchange information during the wireless device pairing operation. In this work, device pairing schemes utilizing out-of-band channels – including audio-based key exchange and device vibrations – were investigated and found to be vulnerable to acoustic eavesdropping attacks.

Additional methods to extract keyboard input focus on other sources of information (i.e., other than audio). In [4], Balzaroni et al. explored recovering keyboard input based on video of the typing session. In this approach, the manual typing is recorded using a video camera and each typed key is assigned a list of possible characters. Then, a language model and context-sensitive techniques are used to choose the best matching characters for each key typed. In [16], Song et al. showed that timing information of key-presses can be used to exploit weaknesses in SSH protocol. The algorithm uses SSH data to first train an HMM model for timing analysis. The system is then used to recover passwords consisting of 5-8 characters.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we took a fresh look at the vulnerability of keyboard typing to audio emanations. Our work shows that keyboard eavesdropping is affected by a few variables, including the typing style, the input data and the detection technique. We showed that while the detection performance is reduced for realistic typing styles, keyboard typing still remains vulnerable to eavesdropping attacks.

Our work further provides an objective measure for the performance of key detection. This information is useful for implementing future language model and dictionary based attacks as their success relies on the underlying audio-based (raw) key detection capability. Further, our work helps asses the performance of these attacks, by providing an estimate as to how much the use of a language model or dictionary may further improve the final detection results.

Overall, we found that the strength of acoustic eavesdropping attacks is 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, we define a Best Guesses Search, which reduces by half the entropy of the typed random passwords and therefore considerably speeds-up the exhaustive search.

There exist several avenues for future work. First, our work concentrated on English alphabets but 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, have the same size and share the same underlying physical plate, we expect the detection behavior to be the same. However, it would be interesting to verify this in future research. Another possible extension can be to look at the combination of the Shift key with other characters. This scenario is interesting since an overlap is expected between the acoustic emanations of the keys which may make it harder to detect the pressed keys.

We believe that testing laptop keyboard acoustic emanations is also an interesting further step. We conducted preliminary tests and noticed that the press signal is evident in laptop keyboard recordings. However, we found that 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.

Acknowledgements

We thank Avishai Wool, Yigael Berger and Doug Tygar for discussions related to prior work on keyboard acoustic emanations.

9. REFERENCES

[1] A. Adams and M. A. Sasse. Users are not the enemy. *Commun. ACM*, 42(12): 40–46, 1999.

[2] D. Asonov and R. Agrawal. Keyboard acoustic emanations. In *IEEE Symposium on Security and Privacy*, 2004.

[3] M. Backes, M. Dı́urmuth1, S. Gerling1, M. Pinkal3, C. Sporleder Acoustic Side-Channel Attacks on Printers. In *Usenix Security Symposium*, 2010.

[4] D. Balzarotti, M. Cova, G. Vigna ClearShot: Eavesdropping on Keyboard Input from Video In*Proceedings of the 2008 IEEE Symposium on Security and Privacy*, 2008.

[5] Y. Berger, A. Wool and A. Yeredor, Dictionary Attacks Using Keyboard Acoustic Emanations. In *Conference on Computer and Communications Security, SESSION: Attacks and cryptanalysis, Pages: 245 - 254*, 2006.

[6] R. Briol Emanation: How to keep your data confidential. In *Symposium on Electromagnetic Security For Information Protection, SEPI*, Nov. 1991.

[7] A. H. Y. Fiona, Keyboard Acoustic Triangulation Attack. Final Year Project, Available at `http://personal.ie.cuhk.edu.hk/~kwwei/FYP/keyboard_acoustic_attack/Eric_Thesis2_final.pdf`

[8] T. Halevi, N. Saxena 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.

[9] P. Inglesant and M. A. Sasse. 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*, pages 383–392, 2010.

[10] A. Moore, School of Computer Science, Carnegie Mellon University. Hidden Markov Model. http://www.autonlab.org/tutorials/hmm14.pdf.

[11] R. Morris and K. Thompson. Password security: a case history. *Commun. ACM*, 22(11):594–597, 1979.

[12] L. Rabiner and B.H. Juang. Mel-Frequency Cepstrum Coefficients. Prentice-Hall Signal Processing Series, 1993, ISBN:0-13-015157-2.

[13] A. Shamir and E. Tromer. Acoustic cryptanalysis: On nosy people and noisy machines. `http://people.csail.mit.edu/tromer/acoustic/`.

[14] R. Lachlan, Normalization for Dynamic Time Warping. `http://luscinia.sourceforge.net/page26/page14/page14.html`.

[15] "Keyboard Acoustic Emanations Revisited" presentation. `http://cs.unc.edu/~fabian/courses/CS600.624/slides/emanations.pdf`.

[16] D. Song, D. Wagner, and X. Tian. Timing analysis of keystrokes and timing attacks on ssh. In *Tenth USENIX Security Symposium*, 2001.

[17] Typing. Wikipedia, available at `http://en.wikipedia.org/wiki/Typing`.

[18] L. Rabiner and B. Juang. Fundamentals of Speech Recognition. In Prentice-Hall, Inc, 1993.

[19] R. Shay, S. Komanduri, K.G. Patrick, P. G. Leon, M. L. Mazurek, L. Bauer, N. Christin and L. F. Cranor. Encountering stronger password requirements: user attitudes and behaviors In SOUPS '10: Proceedings of the Sixth Symposium on Usable Privacy and Security, 2010.

[20] A. Wool and Y. Berger. Personal communication on the subject of typing styles used in prior research on keyboard acoustic emanations. April, 2010.

[21] J. Yan, A. Blackwell, R. Anderson, and A. Grant. Password memorability and security: Empirical results. *IEEE Security and Privacy*, 2(5):25–31, 2004.

[22] L. Zhuang, F. Zhou, J. D. Tygar, Keyboard Acoustic Emanations Revisited. In *Proceedings of the 12th ACM Conference on Computer and Communications Security*, November 2005, pp. 373-382.

[23] L. Zhuang, F. Zhou, J. D. Tygar, Keyboard Acoustic Emanations Revisited. In *ACM Transactions on Information and System Security (TISSEC)*, October 2009, Volume 13 Issue 1, pp. 3-26.

APPENDIX

A. ADDITIONAL TABLES

Table 6: Single Character Detection Rates, 2-character guess

Training Stage →	Hunt and Peck		Touch Typing
Testing Stage →	Hunt and Peck	Touch Typing	Touch Typing
Cross-Correlation	64.89%	43.78%	61.78%
Time-Frequency	78.44%	53.33%	72.67%
Random Guess	3.84%		

B. ADDITIONAL FIGURES

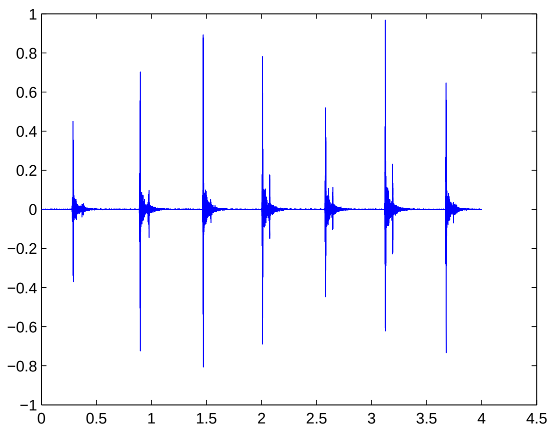

Figure 4: Recording of Multiple Keys

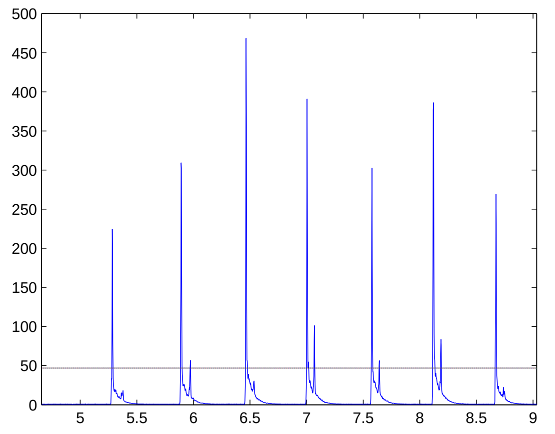

Figure 5: Sum of FFT Coefficients

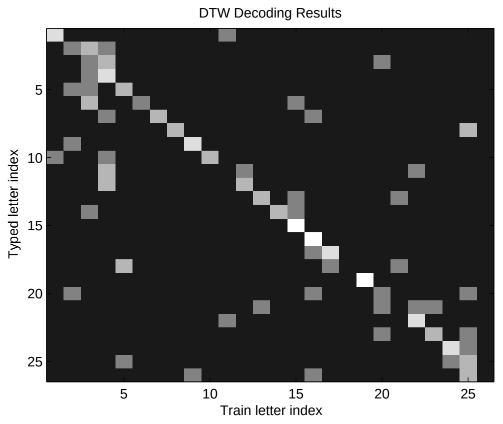

Figure 6: DTW Decoding (4 recordings of each test alphabet letter)

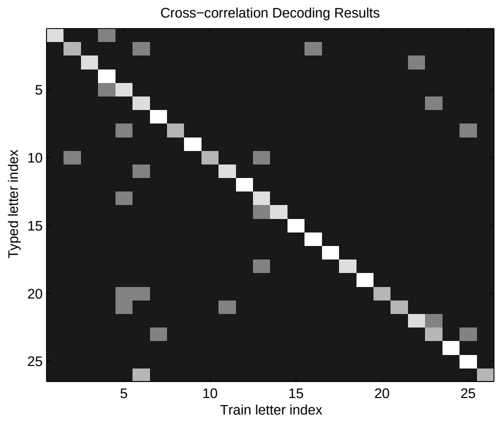

Figure 7: Cross-Correlation Decoding (4 recordings of each test alphabet letter)

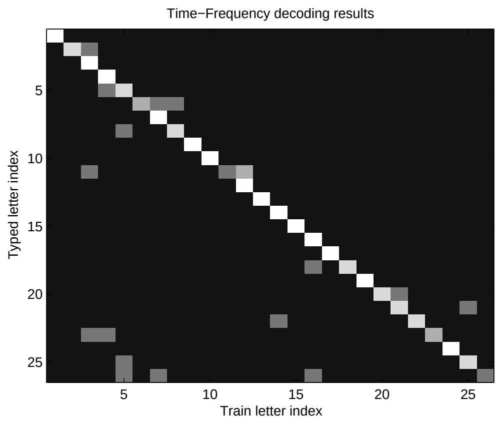

Figure 8: Time-Frequency based Decoding (4 recordings of each test alphabet letter)