A Closer Look at Keyboard Acoustic Emanations: Random Passwords, Typing Styles and Decoding Techniques

ABSTRACT

We take 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. 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. 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, to touch typing, 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 under realistic 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 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 [1] 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. [21, 22] improved upon the attack of [1] by obviating the need for a labeled training recording. Instead, HMM English language-based model [9] was used on a 10-minute typed English text for unsupervised training and labeling of the data (using neural networks and Mel Frequency Cepstrum Coefficients (MFCC)).

Berger et al. [4] further utilized dictionary attacks to decode 8 letter or longer English words utilizing correlation calculations. Their work showed that keys which are in close physical proximity on the keyboard typically have higher cross-correlation than farther ones.

In this paper, we take a fresh look at keyboard acoustic attacks and aim to address some important aspects that prior work did not cover or fully explore. First, we systematically investigate 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. Users are often instructed, and at times forced, to use random passwords [8, 18]. 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]

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 (and similarities between separate clicks of the same key). 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 only used the "hunt and peck" or "search and peck" technique [14, 19]. In this technique, the typist presses each key individually [16]. However, in real-life scenarios, many people employ "touch typing" [16].

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, 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.

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

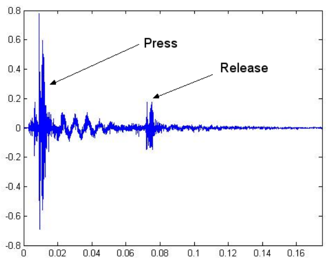

Figure 1: Acoustic Signal of a Single Key

2. THREAT MODEL

Our attack model is very similar to the one considered by prior research on keyboard acoustic emanations [1, 21, 22, 4]. 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. In our experiments, we consider random passwords consisting of lower-case English alphabets.

Our attack examines in-depth the advantage which an adversary can obtain by comparing previously taken recordings of known data to new samples of data. 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 discussed in section 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, our training data is captured in a "mechanical" style (discussed in Section 4.1).

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 for a limited amount of time. The attacker is usually allowed to make a certain number of password trials. We suggest a method of password exhaustive search that reduces significantly the overall search space while increasing the probability of correct password detection.

Attack Set-Up and Tools: Throughout our experiments, we used a standard Lenovo keyboard (model JME7053 English) for our typing, a Logitech USB PC microphone, the Recordpad (V.3.0.3) and Matlab software. The random password characters were generated using the Matlab script "char('a' + ceil(rand(1,6) * 26) - 1)."

3. DETECTION TECHNIQUES

To develop our attack algorithms, we started by exploring techniques 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 (Figure 1), as demonstrated in [1]. Our experiments showed that depending on the force sustained while pressing the key, both the push and the release have between 1 to 3 distinct peaks. We tried using different regions sizes for detecting the correlation between same key presses and found that the best results were obtained for regions of 50 ms.

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 (utilizing the Matlab "specgram" command) the Fast Fourier Transform (FFT) 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) of the appendix. For detecting the key release (which is quieter), we repeat the process using a smaller window of 88 samples.

3.2 Existing Techniques

Dynamic Time Warping: Dynamic Time Warping (DTW) is an algorithm which measures similarities between sequences. We used the simple distance measure between each two elements in the signal vectors to calculate the difference between each two recordings. We implemented the DTW function using C source code for a Matlab executable (MEX)

We experimented using signal normalization based on amplitude, mutual joint distribution [13], and no normalization. We found that energy-based normalization produced the best results. We tried using only the push, release or mean of both and found that the latter provided the best results.

Letter Data Set We created a dataset for each letter. 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 sample with an alphabet key, we calculate the average distance between the test character and the Letter Data Set of each alphabet key and pick as the best match the one with the smallest value.

Cross-Correlation: We performed the cross-correlation (denoted X-Corr) between the recorded signals as done in [4]. 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, we took its Letter Data Set and calculated the average of their cross-correlation measurement with the test sample, receiving one similarity measurement for each key. The matching alphabet letter was chosen as the one with the highest similarity.

Frequency-based Distance Measure: We perform Frequency-domain Features-based Distance Measure (denoted Freq-Dist) similar to the one described in [4]. We compute the frequency-based distance between each two signals by calculating the Euclidean difference between the features for the press and release parts and average them to get a distance measure. We calculate the distance between each test sample and each alphabet Letter Data Set and chose the letter with the smallest distance.

Frequency Features and Neural Networks: We implemented the frequency-domain features based technique, using MFCC features as input to neural networks, as described in [22]. We used 10 ms windows and an 2.5 window step size , computing 13 MFCC per window, and examined a total of 40 ms of each press. As per the original implementation, we utilized Matlab's $newpnn()$ function 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 (Section 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, we used only four instances per alphabet letter for training.

Table 1: Single Character Detection

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

We tested the key detection rate (out of 26 alphabet letters) using each technique.

We found that the cross-correlation technique gave the best results, with a single key detection rate of 73%. The detection results can be found in Table 1. 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, and attempting to wrap the signals reduces the differences between clicks of different keys.

3.4 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 first calculate the frequency distance measure F for each instance, (section 3.2) and X-Corr similarity measurement C.

To combine both elements, we first define the correlation-based distance $DC = 1 - C$ (so both F and DC are ascending).

We examined a few methods of combining both matrices. We tried picking the minimum of each value $(\min(DC, F))$ and the average of the two values (DC, F). We also looked at (F, DC) 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), We further define the time-frequency similarity measure as $S_{TF} = 1 - TF$ and chose the alphabet letter with the highest similarity to each test sample.

Using the time-frequency classification technique, we get an increased probability of 83% for the training data. We therefore conclude 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, we create three datasets.

Straw Man Approach: Our 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 [1]). This causes the finger to hit the key from a vertical position in each case. The benefit of using this technique is that it and 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 ([1]) 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.

We used the above technique to take ten signal recordings for each key of the alphabet letters as our 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, 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.

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 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). We refer to this data as the "Test Touch Typing data".

4.2 Typing Style and Signal Correlation

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

Our training data included 10 training samples using the straw man typed dataset (Section 4.1).

Straw Man Typing: We started by using the aforementioned data as test data itself. For each sample, we calculate the maximum correlation with each of the other instances taken with the same key (termed *matching 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 repeat this calculation for the sample with the Letter Data Sets of the rest of the keys (termed *non-matching keys*). 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 highest correlation for the sample belongs to the Letter Data Set of the matching alphabet key, 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. 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 Straw Man Dataset,

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%

we found that 56% of the press signals and 67% of the release signals best matched their corresponding typed letter.

Hunt and Peck Typing: We next repeat the correlation calculations between the hunt and peck passwords test dataset and the Straw Man initial dataset. We found that the Match probability was reduced to 28% for the press 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. Therefore, 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: 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 release.

A summary of the results is presented in Table 2. In conclusion, we 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 our 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 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 and compare the detection rates 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 which reduces the effect of noise (as opposed to four instances used in Section 4.1).

We calculate the similarity measure – max correlation for cross-correlation and S_{TF} similarity for time-frequency classification – between each tested instance and each alphabet key letter in the training data. For each technique, we chose as the best matching letter the one with the highest similarity measurement.

5.1 Initial Dataset, Straw Man Approach

The initial data is typed with the Straw Man Approach (as discussed in section 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, we found that the cross-correlation statistics calculated using this technique resulted in a 83% accuracy rate per key (since we now use ten keys per training, this raised the result up from 73% when only four training instances were used). When using the time-frequency based classification, we found that the results further improved (to 89%). We conclude that when the typing is repetitive, the underlying physical characteristics of the keyboard has strong effect on the acoustic emanations and the ability to eavesdrop is relatively high.

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%		

5.2 Test Data, Hunt and Peck Style

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

We find that the cross-correlation performance is reduced to a 54% accuracy rate per key. We found that utilizing the time-frequency technique improved the detection rate to 65% per character. We conclude 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.

5.3 Test Data, Touch Typing Style

We repeated the testing process for the passwords typed with the touch typing style (using the straw man dataset as our training data). We find that utilizing the cross-correlation technique for key detection, the accuracy rate is reduced to 34%. For the time-frequency based classification, we observe that the rate of detection per correct character has increased to 41%.

5.4 Best Guesses Search

To raise our detection rate, we tried 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 and a list of keys with the lowest TF distance from the test character. 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 letter.

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

Table 5: 6-Character Password Detection Rates

Method →	EXHAUSTIVE SEARCH						BRUTE FORCE
	Cross-Correlation			**Time-Frequency**			
Training Stage →	Hunt & Peck		Touch	Hunt & Peck		Touch	
Testing Stage →	Hunt & Peck	Touch	Touch	Hunt & Peck	Touch	Touch	
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%

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 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 increased to 88%. For the touch typing data, the rate increased to 75%. Our Results (for single key and five key guesses) are summarized in Tables 3 and 4, respectively. The results corresponding to the two best guesses are depicted in Table 6 of the appendix. (see Appendix).

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 use the typed signals to create the training data. We then compare using 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 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 reduces significantly the computing complexity and speeds-up the attack. Therefore, it can be used when limited-time access is available to a device 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 (a common scenario).

For the Best Guesses Search, we chose for each character the closest five keys. 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 to:

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

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

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

We 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 test the Best Guesses Search detection probability for 6-character passwords (which includes $5^6 = 15625$ trials) and verify that it indeed matches our calculations using Equation 3.

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. We 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 brute force attack (which would produce on average 0.005% success rate for the size of our search space).

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 results for passwords of length up to 12 characters in Figures 2 and 3 of the appendix. We see that the Best Guesses Search significantly reduces the search space size and improves the detection probability for passwords of different length.

6. DISCUSSION

Our 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

We explored several techniques, including DTW, time-based correlation and frequency features. Our 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. We observe that the similarities in signals emanated from the same key are detectable both in the frequency and in the time domain. We present a new technique which combines this information and achieves improved detection results based on both time and frequency data.

6.2 Typing Style

Our work demonstrates that while the underlying plate contributes to the key sound, the typing style also contributes to it significantly. One of our observations is that while there are still sound differences between some of the keys (which confirms our perception that some keys sound "different"), when examining all the alphabet keys in the keyboards, it becomes hard to distinguish between a single key and all the other alphabet keys.

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). 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

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 [21, 22, 4]. 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 random 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 [5], Briol showed that significant information can be extracted about the printed text, using acoustic emanations to distinguish between the letters 'W' and 'J'. In [2], Backes et al. presented an attack which recovers English printed text from the printer audio sounds.

In a proof-of-concept work published on the web [12], Shamir and Tromer explore inferring of CPU activities associated with RSA decryption via acoustic emanations to learn the RSA private keys.

In [7], 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 [3], Balzaroni et al. explored recovering keyboard input based on video of the typing session. In [15], Song et al. showed that timing information of keypresses can be used to exploit weaknesses in SSH protocol. In [23], accelerometer data from modern mobile phones (iPhone 4) was used to implement keyboard dictionary attacks.

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: typing style, input data and detection technique. We show 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 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, 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.

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, we expect the detection behavior to be the same. However, it would be interesting to verify this in the 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. We conducted preliminary tests and noticed 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.

Acknowledgements

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

9. REFERENCES

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

[2] M. Backes, M. Dİurmuth1, S. Gerling1, M. Pinkal3, C. Sporleder Acoustic Side-Channel Attacks on Printers. In *Usenix Security Symposium*, 2010.

[3] 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.

[4] 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.

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

[6] 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`

[7] 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.

[8] 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.

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

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

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

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

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

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

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

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

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

[18] 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.

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

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

[21] 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.

[22] 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.

[23] P. Marquardt, A. Verma, H. Carter, P. Traynor,)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

APPENDIX

A. ADDITIONAL FIGURES AND TABLES

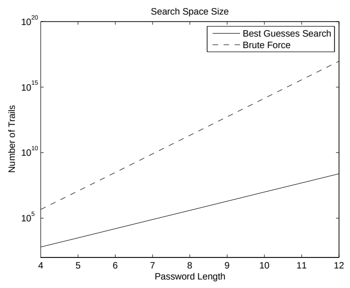

Figure 2: Best Guesses Search space size

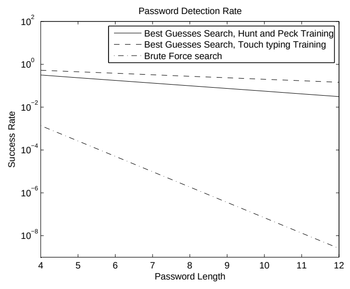

Figure 3: Best Guesses Search detection probability

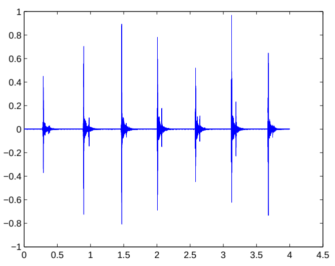

Figure 4: Recording of Multiple Keys

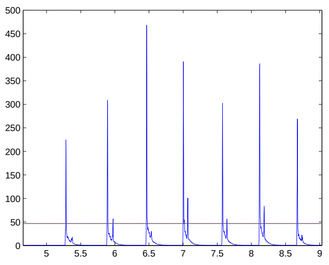

Figure 5: Sum of FFT Coefficients

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%		