Keyboard Acoustic Emanations Attacks

Michael Farrell, Kevin Yang
{michaelfarrell, kyang01}@g.harvard.edu

***Abstract*—We examine how to reconstruct typed text from unlabeled keystroke sounds based off of Zhuang et al.'s work. We replicate their results with a character accuracy rate of 50.1% using only unsupervised techniques. These unsupervised techniques include K-Means clustering and Hidden Markov Models. However, these results are recreated under ideal circumstances where the environment is quiet and the individual types slowly. After exploring the feasibility of keyboard acoustic emanation attacks, we explore how to defend against them. By adding random noise at a volume equal to the standard deviation of volume for keystrokes, we find that we can reduce the character accuracy rate to 27.1%. This implies that random noise can help an individual defend against these attacks when typing sensitive information.**

I. Introduction

Emanations from electronic devices has emerged as a prevalent side channel for attacks. Recent work has shown that acoustic emanations, vibration emanations and optical emanations can all leak valuable information to the outside user. A recent example of a side channel attack uses keyboard acoustic emanations during a video conference to infer typed text [3]. Another relevant example of a side channel attack leverages accelerometer data on smartphones to infer passwords [6].

In this paper, we examine the feasibility of launching a keyboard acoustic emanation attack. We reimplement Zhuang et. al's work on acoustic emanation attacks and examine defenses against these attacks [1]. Using simply an audio recoding of a keyboard typing, Zhuang et al. are able to achieve character accuracy rates of 92% [1]. Because they use unsupervised models, Zhuang et al. does not require a labeled training sample [1]. Zhuang et al. also claims that their attacks work under noisy environments as well [1]. This suggests that these types of keyboard acoustic attacks are highly practical. Using the microphone of a laptop or smartphone to record keystroke can permit an attacker to decipher the text typed including passwords. Given the effectiveness of this technique as shown by Zhuang et al. and the wide prevalence of microphones, keyboard emanation attacks are particularly potent.

With how easily these attacks can be conducted, we explore the possibility of defending against them. In particular, we focus on disrupting the ability of unsupervised classification models. Generating high character accuracy rates using only unsupervised techniques is critical because their predictions can serve as the labeled data for supervised models. We accomplish this by adding random noise to a recording of typing. We illustrate that this technique helps reduce the feasibility of these keyboard acoustic emanation attacks.

II. Related Work

Asonov and Agrawal were the first to illustrate the potential of keyboard acoustic emanation attacks [2]. In their work, they use a recording of keystrokes and their labels. They run FFTs over the labeled keystrokes and use the coefficients as features [2]. Afterwards, Asonov and Agrawal run supervised techniques over these FFT coefficients and train on identifying the labels of the keystrokes [2]. Though they demonstrated the feasibility of keyboard acoustic emanation attacks, Asonov and Agrawal's had many limitations. Their attack required a labeled dataset of the keystroke sounds. In practice, this type of a labeled dataset would be incredibly difficult to capture. Additionally, their trained models were not easily transferable from one keyboard or user to another. Accuracy rates dropped below 25% when running a trained classifier on a different keyboard or user. These drawbacks limited the feasibility of using keyboard emanation attacks in practice.

Zhuang et al.'s increased the feasibility of these attacks by illustrating how a classifier could be reliably trained without a labeled dataset [1]. In their work, Cepstrum feature coefficients were used instead of FFT coefficients to increase prediction accuracy. Furthermore, training the classifier for learning keystrokes was broken into two phases [1]. During the first phase, their attack used K-Means clustering and Hidden Markov Models to assign labels to keys [1]. Using these unsupervised techniques, Zhuang et al. was able to get to character accuracy rates of 60% [1]. After this, they apply spelling corrections and grammar checking to the newly generated rates [1]. This yields a character accuracy rate of 68% [1]. In the second phase of training, they take the Cepstrum coefficients and the labeled characters generated by previous unsupervised classifier and train regression models to predict characters. After repeatedly applying these two phases of unsupervised and supervised training in a feedback loop, Zhuang et al. are able to achieve classification rates of 96% [1].

Berger et al. show that keyboard acoustic emanation attacks work under limited data circumstances if the individual is typing English words [5] With less than < 5 second recording of typed words, he shows that they can determine the correct word out of 50 from with a 90% accuracy rate [5]. The use similarity metrics based on keystroke similarity to generate constraints for keystrokes. Based off of the series of similar keystrokes, they prioritize a list of likely words. Their similarity analysis on keystrokes relies on the existence of repeated letters in the words and improves with the length of the word. By demonstrating that words can be identified with a recording of less than 5 seconds, they discuss how their results can be used as an acoustic-based dictionary password cracker that improves in performance if the password is longer.

In contrast, Zhu et al. approaches this problem without leveraging knowledge of English words [7]. They argue context based approaches make assumptions that are frequently not valid like in the case of passwords where words are often jumbled characters [7]. Their approach involves placing multiple standard phones around the individual who is typing. Afterwards, they examine the acoustic signals and their time of arrival across each phone. Based on different times of arrival, they find that they can recover 72.2% of keystrokes accurately [7].

More recently, Compagno et al. have illustrated that these types of attacks can also occur over Voice-over-IP (VoIP) software [3]. Their attack is motivated by the observation that most people also engage in typing while on VoIP calls [3]. These keystrokes can be recorded and transmitted over the call. In their paper, they show that Skype transmits enough audio to train a classifier to decipher keystrokes typed on the keyboard. If the attacker is familiar with the keyboard and typing style, the attacker can guess a group of 5 characters that the key is in with an accuracy of 91.7% [3].

III. Building the Attack

To examine defenses against keyboard emanation attacks, we first needed to be able to launch acoustic emanation attacks. Consequently, we reimplemented unsupervised classifiers from Zhuang et al.'s paper.

A. Building the Dataset

To create a dataset to train on, we typed the beginning of Carroll's Alice in Wonderland for 20 minutes on a MacBook Pro. While typing, we tried to minimize external noise by typing in a quiet room. Similar to Zhuang et al. we avoided using the shift and backspace characters. Over the 20 minutes, we typed 2751 characters. A snippet of the dataset and the corresponding signal is shown below.

```
alice was beginning to get very tired of
sitting by her sister on the bank and of
having nothing to do...
```

B. Detecting Peaks

Given the sound data, we wanted to distinguish between silence and individual keystroke. Experiments from Asonov and Agrawal suggest that the average period between keypresses is greater than 100 milliseconds [2]. Each keystroke lasts around 40ms and consists of both a touch peak and a hit peak. The touch peak corresponds to the sound when the finger touches the key, and the hit peak corresponds to the key hitting the supporting key plate. An example keystroke is shown below.

To detect these keystrokes, we closely followed the steps of Zhuang et al.'s work. We applied windowed discrete-time Fourier transforms to the sound signal. Afterwards, we summed over the signal FFT coefficients from 400 Hz to 12000Hz for each time step. Next, we ran peak signal detection libraries to determine where likely keystrokes were. For these peak signal detection libraries, we varied the minimum spacing between consecutive peaks and the minimum threshold until we achieved reasonable results.

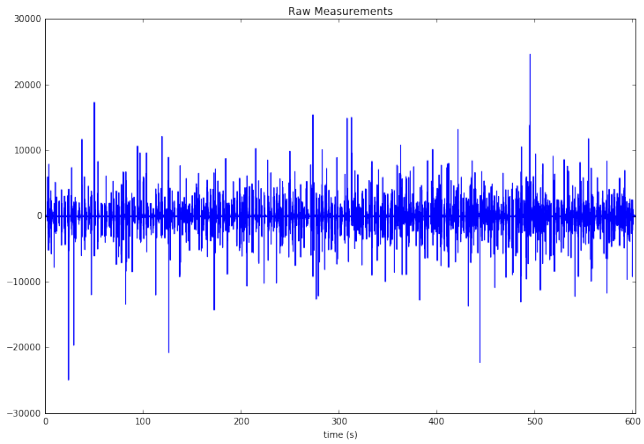

Fig. 1. Example of the raw sound data of typing Alice in Wonderland. The variability in loudness of each keystroke makes the task of classifying text much more difficult.

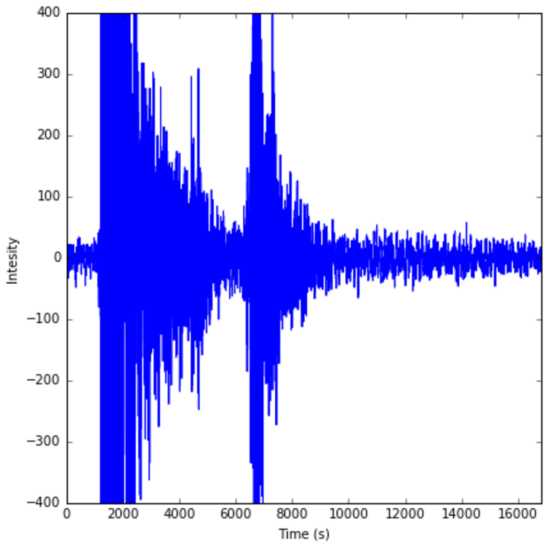

Fig. 2. Examples of signal for the space keystroke.The first peak is the touch peak when the finger touches the key and the second peak is the hit peak that corresponds to the key hitting the supporting key plate.

To get the best possible results, we needed to make corrections by listening to both the recording and our predictions. Corrections in this procedure were also made by Zhuang et al. We built a system for playing both the recording and our predictions in unison to make sure that all keystrokes were annotated. Finally, we also noted the positions of keystrokes that were likely spaces according to their sound. Zhuang et al. also manually annotated the data with spaces to help clustering perform better. Data on the location of spaces helps improves accuracy because it more accurately breaks down the sequence of text into words.

C. Extracting Cepstrum Features

After finding these peaks, we generated Cepstrum features, which were shown by Zhuang et al. to generate better results than FFT features. To generate these features, we closely followed Zhuang et al.'s approach. For each keypress, we generated a window of 80ms to cover both the touch peak and the hit peak. We computed Mel-Frequency Cepstrum Coefficients (MFCC) for each window of 10ms shifting 2.5ms each time. While computing these MFCCs, we set the number of channels in the Mel-Scale Filter to 32 and used the first 16 MFCCs for each window. This resulted in a vector of dimension 1200 for each of the 2751 keypresses we found.

D. Clustering with K-Means

Because of the high dimensional nature of the data, we applied Principal Component Analysis to reduce the dimension of each keystroke vector to 50. Afterwards, we applied K-Means to cluster different sounds together. K-Means++ is a clustering algorithm that breaks data into K clusters where each observation is part of the cluster with the closest mean. We normalized the data and ran K-Means++ to cluster the keystroke vectors into $K = 50$ clusters or effectively 50 different labeled keystroke sounds.

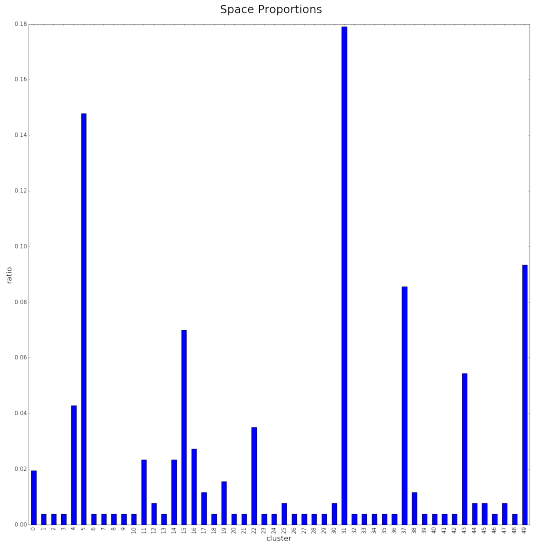

Fig. 3. Examples of the labeled clusters that the space key falls under after running K-means.

E. Building a Hidden Markov Model

Afterwards, we set up a hidden Markov model (HMM) to predict a hidden sequence of characters that could have generated these labeled sounds. A hidden Markov model consists of X possible hidden state variables that produce O different emissions. The HMM assumes that the Markov property is true. This property states that the hidden state variable only depends on the previous hidden state variable. Additionally, each hidden state variable follows a categorical distribution with different emission probabilities. HMMs are trained using the expectation-maximization algorithm and decoded into the most probable sequence of hidden states using the Viterbi algorithm. An example diagram of HMM is shown below.

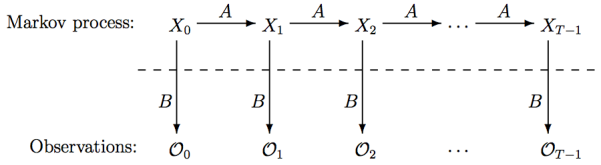

Fig. 4. Example diagram of an HMM with hidden states X and emissions O. Transitions between hidden states are represented with A and transitions from hidden states to emissions is represented with B. Figure taken from from *Stamp's A Revealing Introduction to Hidden Markov Models* [4]

In our case, each of the 27 possible characters is a hidden state and each of the 50 possible labeled sounds is a potential emission. Using a HMM assumes that the keystroke for each character has a unique distribution of possible sounds. The HMM also makes the simplifying assumption that each character depends only on the previous character and the sound generated. The sequence decoded by the Viterbi algorithm is the most probable sequence of characters given the sequence of keystroke sounds. While training the HMM, we held the transition matrix between characters fixed. The transition matrix was initialized to the relative bigram frequency across 20 books from Project Gutenberg. Training and decoding with the HMM algorithm only took a few minutes on a 2014 Macbook Pro.

F. Parameter Tuning

Extensive parameter tuning was also applied for running principal components analysis, K-Means clustering, and the Hidden Markov Model. For principal components analysis, we tuned the number of components to use from 30 to 100. For K-Means clustering, we tuned the number of clusters to use from 45 to 60. Finally, for the Hidden Markov Model, we varied the amount of smoothing we did for our transition matrix and the spaces vector we initialized the HMM with. We manually tuned these parameters until we began to see words we recognized.

G. Evaluation

We evaluated sequences we found against the true typed sequence of Alice in Wonderland. Our evaluation metric for comparing models was character accuracy. This is the ratio of matching characters at the same position between the predicted and true sequences divided by the total number of characters in the true sequence.

IV. Defenses

After building models for deciphering a audio recording of keystrokes into a sequence of most probable characters, we wanted to explore how to defend against these types of acoustic emanation attacks. In particular, we wanted to explore how much white noise would need to be added to thwart these types of attacks.

A. Adding Random Noise

Random noise was layered on top of an existing audio recording by using the python wave package. This noise was generated with a standard deviation equal to the raw signal of keystrokes. Afterwards, the random noise was scaled by a constant ratio and added to the keystroke data.

B. Retraining the Classification Model

After adding white noise to the data, we retrained the classification model on this new dataset. We extracted Cepstrum features, clustered using K-Means, and trained a new HMM. Next, we compared classification accuracy results against the previous baselines we generated from noiseless circumstances.

V. Results:

While reimplementing Zhuang et al.'s work, we noticed that text needs to be typed slowly for it to be correctly classified. We show our results on this below.

Words Per Minute	Character Accuracy
21.0	50.1%
37.5	29.4%

We found that character accuracy was highest in a quiet environment where the individual typed slowly. In these situations, we were able to decipher around 50.1% of all characters. An example of the prediction sequence and target sequence is shown below.

```
Target: alice was beginning to get very
tired of sitting by her sister...

Predicted: alice wat hedincond to bed vexp
thred th tintins pr her torthr..
```

In the ideal case where the user typed slowly and the environment was relatively quiet, some words can be deciphered from the predicted chars. For example, we can decipher the words "alice", "was", and "her" in the above example. These words could serve as part of the labeled dataset for the supervised feedback loop.

However, in other cases where the individual typed quickly, the results were undecipherable. With character accuracy rates of 29.4%, the techniques of employing this labeled dataset to serve as a labeled training set for a supervised classification model seems far less feasible. An example of a prediction sequence where we typed quickly is shown below.

```
Predicted: aimbe wat helvering toules beis
threr thicanting mporericorupr
```

Consequently, the results were also particularly susceptible to typing speed. Zhuang et al.'s results of 96% classification accuracy are on recordings where the individual types at around 30 words per minute. Considering that many individuals can type at speeds upwards of 75 words per minute, these attacks become far less effective.

We also found that character accuracy rates suffer when spaces were not correctly identified. Spaces are critical because they help identify the word boundaries. When identified incorrectly, the HMM frequently generates long uninterrupted blocks of undecipherable text. Zhuang et al. also notices this and suggest that spaces should be identified manually to improve results. However, this is problematic because manually identifying spaces in audio recording is difficult. Detecting spaces is especially difficult when the individual is typing quickly.

Finally, we experimented with how classification accuracy is affected by white noise. In their paper, Zhuang et al. claims that their techniques works in noisy environments [1]. However, upon closer analysis of their results, we realize that "noisy" simply means an unexpected cellphone noise and the static background of a computers fan. The noise from the cellphone can be easily tuned out by ignoring the segment of the recording affected. Meanwhile, the static background noise can be easily filtered out because of its consistency and low frequency.

We wanted to examine how the classification model would do when the noise was random and loud. Below, we show some of our results for modifying the volume of the noise up to a standard deviation of the keystroke across $N = 40$ trials.

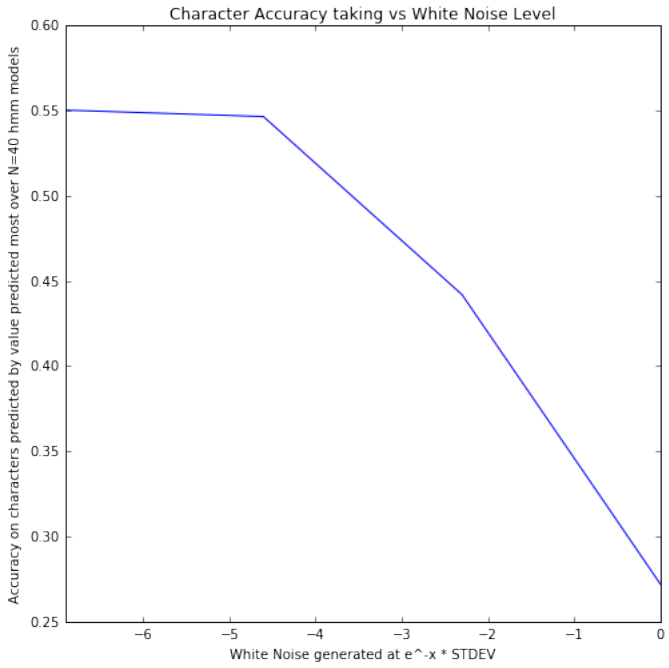

Fig. 5. White noise was added to the keystroke dataset with the volume ranging from 0 to e^-4 times the standard deviation of keystroke volume.

Upon adding white noise equal to the standard deviation of the keystroke volume, the character accuracy rate drops to 27.1% accuracy. This amount of noise is sufficient to result in virtually undecipherable text. Below are additional figures that show the character accuracy rates on recordings with added random noise.

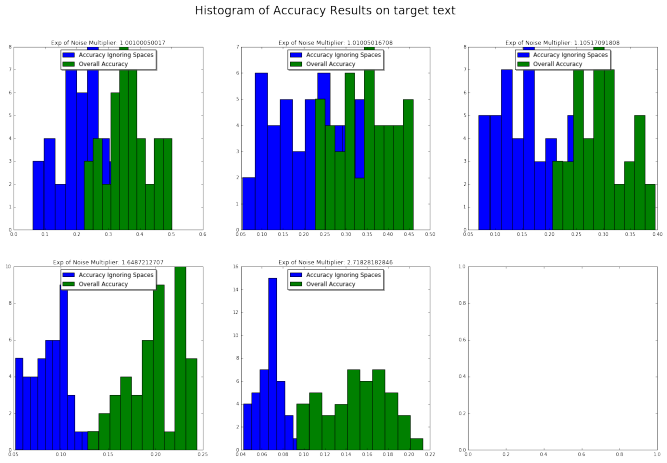

Fig. 6. A histogram of the accuracy from the hmm model. The green plot is the overall accuracy of each individual char, and the blue plot is the accuracy of the characters without spaces. The top left plot has the least noise introduced and the bottom right has the most noise introduced. Note that when minimal noise is introduced, the hmm model will find various local minima.

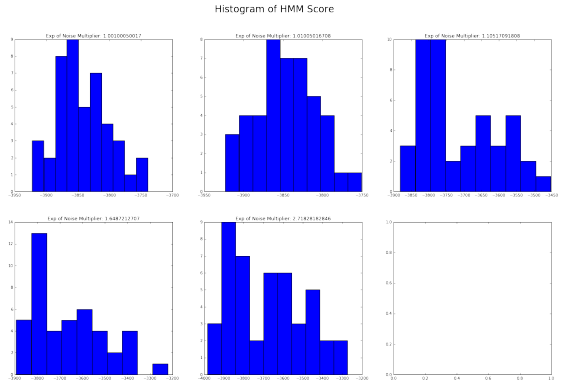

Fig. 7. A histogram of the output score from the hmm model. The top left plot has the least noise introduced and the bottom right has the most noise introduced.

VI. Conclusion:

In our analysis, we focused on reproducing and also disrupting the ability of classifying keystrokes using unsupervised techniques. Feedback methods using supervised techniques rely on the labeled datasets generated by unsupervised techniques. Consequently, by reducing the efficacy of unsupervised techniques, we can reduce the effectiveness of keyboard acoustic emanation attacks.

We achieved character accuracy rates comparable to Zhuang et al's results with a classification accuracy rate of 50.1%. However, we found that the results were not as stable as the authors seemed to suggest. The individual needs to be typing at relatively slow speeds, and spaces need to be accurately identified by manually listening to the recordings. Only under these situations can character accuracies achieve satisfying results with unsupervised techniques.

Our results also show that the potency of these attacks can be further mitigated by playing white noise while typing. Furthermore, we found that character accuracy can be adversely affected by white noise that plays at a volume equal to the standard deviation of the keystrokes. With this, the classification accuracy drops down to an accuracy of 27.1%. This agrees with the intuition that an increase of noise in the keystroke signal will lead to a decrease in the character classification accuracy. Consequently, this suggests that precautions can be taken against these types of acoustic attacks.

These types of limitations imply that these attacks are less practical than Zhuang et al. suggest in his initial paper [1]. Recent work with lower character classification accuracy seems to confirm this [7] [3]. These attacks can only be used under circumstances where background noise is limited, space keys can be easily identified, and the individual types slowly. These constraints make the attack much more difficult to successfully execute.

VII. Future Work:

In the future, we would like to generate methods for automatically detecting spaces in a sound recording. Similar to Zhuang et al.'s results, we found that the accuracy of locating spaces substantially affected the performance of character classification. Training models to automate the detection of the space key helps reduce the amount of manual effort necessary to launch these attacks.

Additionally, we would like to explore incorporating some of Berger's results in using character similarity to build constraints within words. This will allow our system to leverage repeated characters within words and other similarity characteristics within words to help make more accurate predictions.

Finally, we noticed that prior work does not incorporate information about the timing between different keystrokes. We imagine that timing between keystrokes gives information on text typed because it is the result hand to move from one key to the next. Longer intervals of times between keys suggests that the key may be farther away on a QWERTY keyboard like characters "Q" or "P". We believe that this temporal information would dramatically improve the classification accuracy.

References

[1] Zhuang, Li, Feng Zhou, and J. Doug Tygar. "Keyboard acoustic emanations revisited." ACM Transactions on Information and System Security (TISSEC) 13.1 (2009): 3.

[2] Asonov, Dmitri, and Rakesh Agrawal. "Keyboard acoustic emanations." IEEE Symposium on Security and Privacy. Vol. 2004. 2004.

[3] Compagno, Alberto, et al. "Don't Skype & Type! Acoustic Eavesdropping in Voice-Over-IP." arXiv preprint arXiv:1609.09359 (2016).

[4] Stamp, Mark. "A revealing introduction to hidden Markov models." Department of Computer Science S gan Jose State University (2004).

[5] Berger, Yigael, Avishai Wool, and Arie Yeredor. "Dictionary attacks using keyboard acoustic emanations." Proceedings of the 13th ACM conference on Computer and communications security. ACM, 2006.

[6] Owusu, Emmanuel, et al. "ACCessory: password inference using accelerometers on smartphones." Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications. ACM, 2012.

[7] Zhu, Tong, et al. ”Context-free attacks using keyboard acoustic emanations.” Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014.

VIII. Appendix

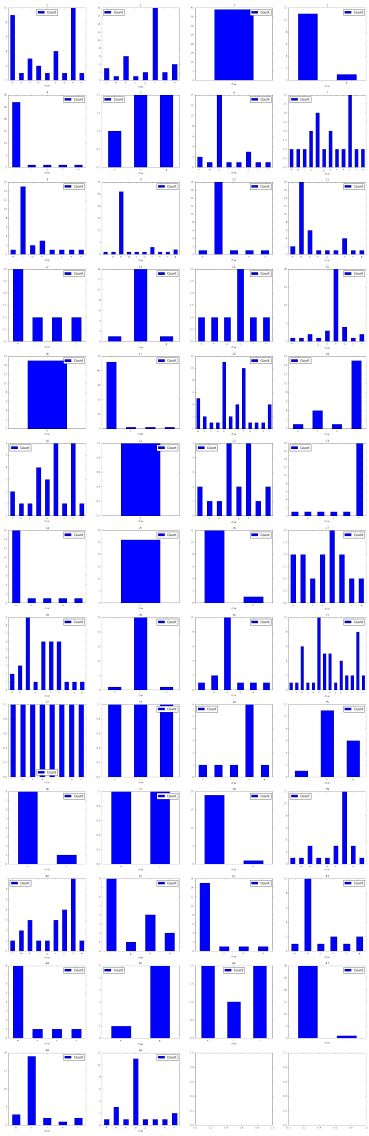

Fig. 8. The histograms of the distribution of characters in each of the k-means buckets. They were clustered soley based off of the frequency of each keystroke.