To appear in the ACM Transactions on Information and Systems Security

Keyboard Acoustic Emanations Revisited

Li Zhuang, Feng Zhou, J. D. Tygar

University of California, Berkeley
{zl,zf,tygar}@cs.berkeley.edu

We examine the problem of keyboard acoustic emanations. We present a novel attack taking as input a 10-minute sound recording of a user typing English text using a keyboard and recovering up to 96% of typed characters. There is no need for training recordings labeled with the corresponding clear text. A recognizer bootstrapped from a 10-minute sound recording can even recognize random text such as passwords: In our experiments, 90% of 5-character random passwords using only letters can be generated in fewer than 20 attempts by an adversary; 80% of 10-character passwords can be generated in fewer than 75 attempts by an adversary. In the attack, we use the statistical constraints of the underlying content, English language, to reconstruct text from sound recordings without knowing the corresponding clear text. The attack incorporates a combination of standard machine learning and speech recognition techniques, including cepstrum features, Hidden Markov Models, linear classification, and feedback-based incremental learning.

Categories and Subject Descriptors: K.6.5 [**Security and Protection**]: Unauthorized access; K.4.1 [**Public Policy Issues**]: Privacy

General Terms: Security

Additional Key Words and Phrases: Computer Security, Human Factors, Acoustic manations, Learning Theory, Hidden Markov Models, HMM, Cepstrum, Signal Analysis, Keyboards, Privacy, Electronic Eavesdropping

1. INTRODUCTION

This paper reports on recovering keystrokes typed on a keyboard from a sound recording of the user typing. Emanations produced by electronic devices have long been a topic of concern in the security and privacy communities [Briol 1991]. Both electromagnetic and optical emanations have been used as sources for attacks. For example, Kuhn was able to recover the display on CRT and LCD monitors using indirectly reflected optical emanations [Kuhn 2002; 2003]. Acoustic emanations are another source of data for attacks. Researchers have shown that acoustic emanations of matrix printers carry substantial information about the printed text [Briol 1991]. Some researchers suggest it may be possible to discover CPU operations from acoustic emanations [Shamir and Tromer 2004]. In ground-breaking research, Asonov and Agrawal showed that it is possible to recover text from the acoustic emanations from typing on a keyboard [Asonov and Agrawal 2004].

Most emanations, including acoustic keyboard emanations, are not uniform across different instances, even when the same device model is used; and they are affected by the environment. Different users on a single keyboard or different keyboards (even of the same model) emit different sounds, making reliable recognition hard [Asonov and Agrawal 2004]. Asonov and Agrawal achieved relatively high recognition rate (approximately 80%) when they trained neural networks with text-labeled sound samples of the same user typing on the same keyboard. Their attack is analogous to a known-plaintext attack on a cipher – the cryptanalyst has a sample of plaintext (the keys typed) and the corresponding ciphertext (the recording of acoustic emanations). This *labeled training* sample requirement suggests a limited attack, because the attacker needs to obtain training samples of significant length. Presumably these could be obtained from video surveillance or network sniffing. However, video surveillance in most cases should render the acoustic attack irrelevant, because even if passwords are masked on the screen, a video shot of the keyboard could directly reveal the keys being typed.

In this paper we argue that a labeled training sample requirement is unnecessary for an attacker. This implies keyboard emanation attacks are more serious than previous work suggests. The key insight in our work is that the typed text is often not random. When one types English text, the finite number of mostly used English words limits possible temporal combinations of keys, and English grammar limits word combinations. One can first cluster (using unsupervised methods) keystrokes into a number of acoustic classes based on their sound. Given sufficient (unlabeled) training samples, a *most-likely mapping* between these acoustic classes and actual typed characters can be established using the language constraints.

This task is not trivial. Challenges include: 1) How can one mathematically model language constraints and mechanically apply them? 2) In the first sound-based clustering step, how can one address the problem of different keys clustered in the same acoustic class and a single key clustered in multiple acoustic classes? 3) Can we improve the accuracy of the guesses by the algorithm to match the level achieved with labeled samples?

Our work answers these challenges, using a combination of machine learning and speech recognition techniques. We show how to build a keystroke recognizer that has better recognition rate than labeled sample recognizers in [Asonov and Agrawal 2004]. We only use a sound recording of a user typing.

Our method can be viewed as a machine learning version of classic attacks to simple substitution ciphers. Assuming the ideal case in which a key produces exactly the same sound each time it is pressed, each keystroke could be easily given an acoustic class according to the sound. The acoustic class assignment would be a permutation of the key labels. This is exactly an instance of substitution cipher. Early cryptographers developed methods for cryptanalyzing substitution ciphers. Our attack can be viewed as an extension of these methods – but our problem is more difficult because the sound of a particular keystroke varies even when it is produced by the same typist.

We built a prototype that can bootstrap the recognizer from about 10 minutes of English text typing, using about 30 minutes of computation on a desktop computer with a Pentium IV 3.0G CPU and 1GB of memory. After the bootstrap step, it could recognize language-independent keystrokes in real time, including random keystrokes occurring in passwords, with an accuracy rate of about 90%. When language-dependent constraints are applied to English text, we achieve a 90-96% accuracy rate for characters and a 75-90% accuracy rate for words[1].

[1]The accuracy rate for words is counted by number of correctly recognized words over total number of words typed.

We posit that our framework also applies to other types of emanations with inherent statistical constraints, such as power consumption or electromagnetic radiation. One only need adapt the methods of extracting features and modeling constraints. Our work implies that emanation attacks are far more challenging, serious, and realistic than previously realized. Emanation attacks deserve greater attention in the computer security community.

Below, Section 2 briefly reviews previous keyboard emanation attacks. Section 3 presents an informal description of the new attack, followed by additional details in Section 4. Section 5 presents experiment results. Section 6 discusses issues and future work. Section 7 concludes the paper.

2. OVERVIEW OF PREVIOUS ATTACKS

We briefly review two related previous research studies examining recovery of keystrokes, each using a different type of side channel information (See [Bar-El 2003] for an overview of side channel attacks in general).

To the best of our knowledge, Asonov and Agrawal were the first researchers to publish a concrete attack exploiting keyboard acoustic emanations [Asonov and Agrawal 2004]. They note that the sound of keystrokes differ slightly from key to key. They give a concrete method to recover information about typing on keyboards, using neural networks as acoustic classifiers. Their approach is to first "teach" the neural networks about what the different keys sound like. To do this, each key is typed 100 times. The neural network is trained with the label (the key being typed) and the corresponding sound. The raw digitalized sound input is too large for their neural networks, so each keystroke is represented as a vector of Fast Fourier Transform (FFT) features. The trained neural network then can be used to recognize subsequent keystrokes.

Based on the *supervised learning* approach above, Asonov and Agrawal show:

—A wide variety (e.g. different keyboards of the same model, different models, different brands) of keyboards have keys with distinct acoustic properties.

—Sound recordings from as far away as 15 meters suffice for neural network supervised learning if sophisticated microphones such as parabolic microphones are used.

—Their neural network supervised learning is sensitive to training errors: if input label are inaccurate, their recognition rates drop sharply. The effectiveness of the appraoch also depends a lot on the comprehensiveness of the training samples, i.e. whether it contains enough samples for each key or not.

Asonov and Agrawal's work opened a new field. However, there are limitations in their approach:

—Their attack is for *labeled* acoustic recordings. Their attack works well only with the same settings (i.e. the same keyboard, person, recording environment, etc.) as the training recording, and such training data are hard to obtain in many cases. Training on one keyboard and recognizing on another keyboard of the same model yields much lower accuracy rates, at around 25%. Even if we count all occasions when the correct key is among the top four candidates, the accuracy

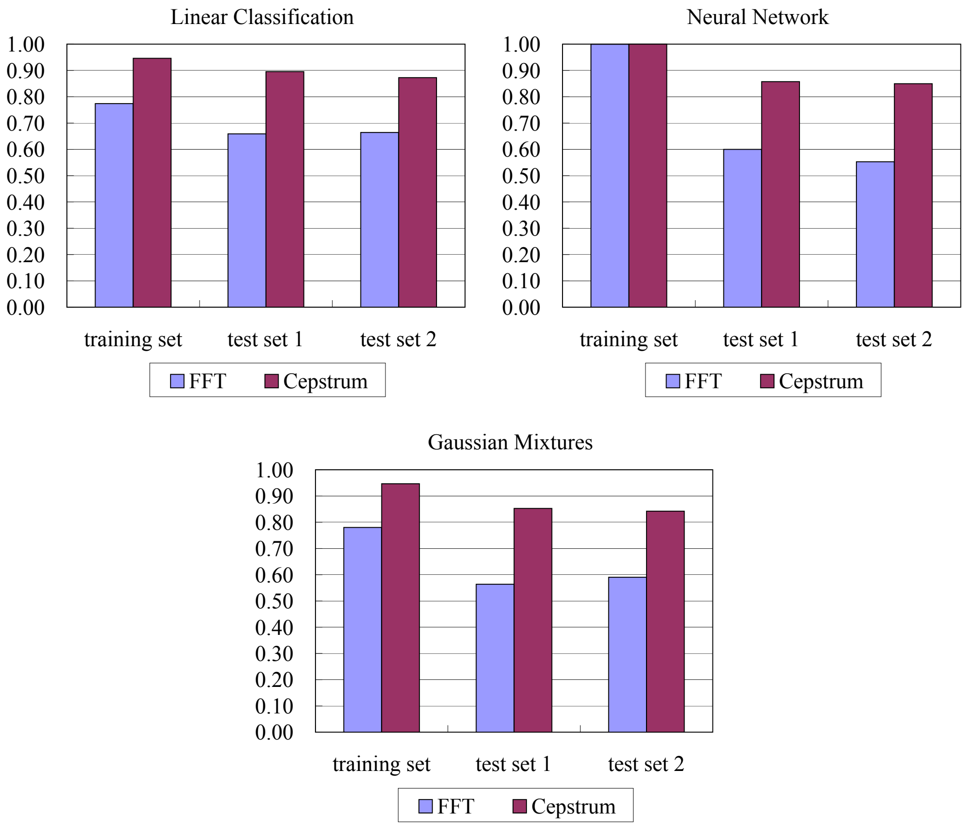

Fig. 1. Recognition rates using FFT and cepstrum features. The Y axis shows the recognition rate. Three different classification methods are used on the same sets of FFT or cepstrum features.

rate is still only about 50%. Lower recognition rates are also observed when the system is trained for one typist and then applied to another typist.

—The set of acoustic classification techniques used leaves room for improvement. In our work, we found superior features to FFT and superior acoustic classifiers to neural networks. Figure 1 compares FFT and cepstrum features and also compares three classifiers: linear classification, neural networks and Gaussian mixtures. The classifier is trained on the *training set* data and is then used to classify the training set itself and two other data sets. Character recognition rate using cepstrum features (discussed below) on average is better than character recognition using FFT. This is true for all data sets and classification methods. Neural networks perform worse than linear classification on the two test sets. In this experiment, we could only approximate the experiment settings in [Asonov and Agrawal 2004]. But the significant performance differences indicate that there are better alternatives to FFT and neural networks combination.

Timing information is a different type of side channel information related to keyboard typing. Timing information includes the time between two keystrokes, the time between keystroke push to keystroke release, etc. Song, Wagner and Tian showed how to extract information based on the time between two consecutive keystrokes [Song et al. 2001]. They considered interactive login shells encrypted with the SSH protocol. In this scenario, an eavesdropper can detect the time between consecutive keys. Statistical analysis shows that the distribution of time between a pair of keys vary for different key pairs. Contributing factors include: whether keys are typed with alternating hands or the same hand, with different fingers or the same fingers, etc. The types of pairs defined in their work capture the physical distances between keys and also the the response time of human beings. However, many different pairs may belong to the same type, e.g. two letters typed by alternating hands. Timing information is generally not helpful in distinguishing different pairs in the same type. Their work gives some analysis of the amount of information leaked by timing information. In Section 6.1, we give an approach to combine timing information with our acoustic emanation recognition. However, to date we have only observed modest improvements by adding timing information. It remains an open question whether the two methods together can yield substantially higher recognition rates.

3. THE ATTACK

In this section, we present a survey of our attack. Section 4 presents the attack in full.

We take a recording of a user typing English text on a keyboard, and produce a recognizer that can, with high accuracy, determine subsequent keystrokes from sound recordings if it is typed by the same person, with the same keyboard, under the same recording conditions. These conditions can easily be satisfied by, for example, placing a wireless microphone in the user's work area or by using parabolic or laser microphones from a distance. Although we do not necessarily know in advance whether a user is typing English text, in practice we can record continuously, try to apply the attack, and see if meaningful text is recovered.

Figure 2 presents a high level overview of the attack.

The first phase (Figure 2(a)) trains the recognizer. It contains the following steps,

—*Feature extraction.* We use cepstrum features, a technique developed by researchers in voice recognition [Childers et al. 1977]. As we discuss below, cepstrum features give better results than FFT.

—*Unsupervised key recognition* using unlabeled training data. We cluster each keystroke into one of K acoustic classes, using standard data clustering methods. K is chosen to be slightly larger than the number of keys on the keyboard. As discussed in Section 1, if these acoustic clustering classes correspond exactly to different keys in a one-to-one mapping, we can easily determine the mapping between keys and acoustic classes. However, clustering algorithms are imprecise. Keystrokes of the same key are sometimes placed in different acoustic classes and conversely keystrokes of different keys can be in the same acoustic class. We let the acoustic class be a *random variable* conditioned on the actual key typed. A particular key will be in each acoustic class with a certain probability. In well clustered data, probabilities of one or a few acoustic classes will dominate for each key. Once the conditional distributions of the acoustic classes are determined, we try to find the most likely sequence of keys given a sequence of acoustic classes for each keystroke. Naively, one might think picking the letter with highest probability for each keystroke yields the best estimation and we can declare our job done. But we can do better. We use a Hidden Markov Model (HMM) [Rabiner and Juang 1986]. HMMs model a stochastic process with state. They capture the correlation between keys typed in sequence. For example, if the current key can be either "h" or "j" (e.g. because they are physically close on the keyboard) and we know the previous key is "t", then the current key is more likely to be "h" because "th" is more common than "tj". Using these correlations in English[2], both the keys and the key-to-class mapping distributions can be efficiently estimated using standard HMM algorithms. This step yields accuracy rates of slightly over 60% for characters, which in turn yields accuracy rates of over 20% for words.

—*Spelling and grammar checking.* We use dictionary-based spelling correction and a simple statistical model of English grammar. These two approaches, spelling and grammar, are combined in a single Hidden Markov Model. This increases the character accuracy rate to over 70%, yielding a word accuracy rate of about 50% or more. At this point, the text is quite readable (see Section 4.3).

—*Feedback-based training.* Feedback-based training produces a keystroke acoustic classifier that does not require an English spelling and grammar model, enabling random text recognition, including password recognition. In this step, we use the previously obtained corrected results as labeled training samples. Note that our corrected results are not 100% correct. We use heuristics to select words that are more likely to be correct. For examples, a word that is *not* spell-corrected or one that changes only slightly during correction in the last step is more likely to be correct than those that had more changes. In our experiments, we pick out those words with fewer than 1/4 of characters corrected and use them as labeled samples to train an acoustic classifier. The recognition phase (Figure 2(b), described below) recognizes the training samples again.This second recognition typically yields a higher keystroke accuracy rate. We use the number of corrections made in the spelling and grammar correction step as a quality indicator. Fewer corrections indicate better results. The same feedback procedure is performed repeatedly until no significant improvement is seen. In our experiments, we perform three feedback cycles. Our experiments indicate both linear classification and Gaussian mixtures perform well as classification algorithms [Jordan 2005], and both are better than neural networks as used in [Asonov and Agrawal 2004]. In our experiments, character accuracy rates (without a final spelling and grammar correction step) reach up to 92%.

[2]Other languages than English have different probabalistic distributions of pairs, but the method still applies.

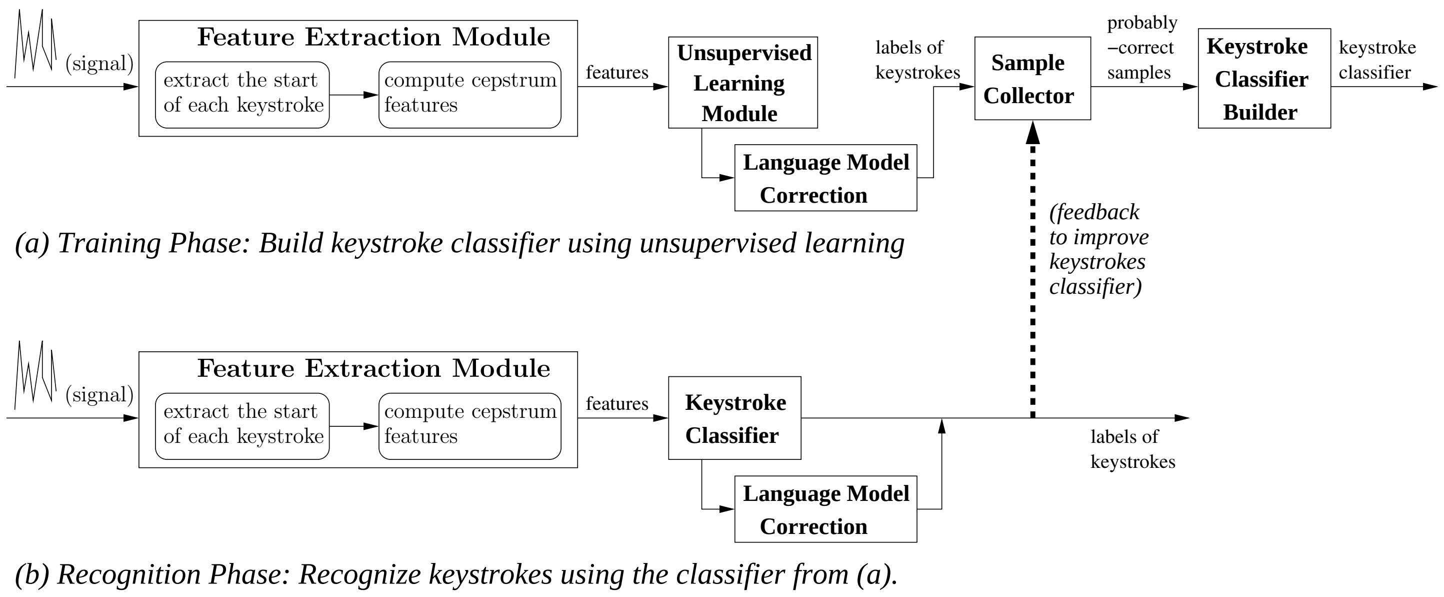
(signal)
Feature Extraction Module
extract the start of each keystroke
compute cepstrum features
features
Unsupervised Learning Module
Language Model Correction
labels of keystrokes
Sample Collector
probably −correct samples
Keystroke Classifier Builder
keystroke classifier
(a) Training Phase: Build keystroke classifier using unsupervised learning
(feedback to improve keystrokes classifier)
(signal)
Feature Extraction Module
extract the start of each keystroke
compute cepstrum features
features
Keystroke Classifier
Language Model Correction
labels of keystrokes
(b) Recognition Phase: Recognize keystrokes using the classifier from (a).

Fig. 2. Overview of the attack.

The second phase, the recognition phase, uses the trained keystroke acoustic classifier to recognize new sound recordings. If the text consists of random strings, such as passwords, the result is output directly. For English text, the above spelling and grammar language model is used to further correct the result. To distinguish between two types of input, random or English, we apply the correction and see if reasonable text is produced. In practice, a human attacker can typically determine if text is random. An attacker can also identify occasions when the user types user names and passwords. For example, password entry typically follows a URL for a password protected website. Meaningful text recovered from the recognition phase *during an attack* can also be fedback to the first phase. These new samples along with existing samples can be used together to increase the accuracy of the keystroke classifier. Our recognition rate improves over time (see below).

Our experiments include data sets recorded in quiet and noisy environments and with four different keyboards (See Table II and Table IV). Refer to Appendix A for an example of recovered text.

4. TECHNICAL DETAILS

Below, we describe in detail the steps of our attack. Some steps (feature extraction and supervised classification) are used in both the training phase and the recognition phase.

4.1 Keystroke Feature Extraction

4.1.1 *Keystroke Extraction.* Typical users can type up to about 300 characters per minutes. Keystrokes consist of a push and a release. Our experiments confirm Asonov and Agrawal's observation that the period from push to release is typically about 100 milliseconds. That is, there is usually more than 100 milliseconds between consecutive keystrokes, which is large enough to distinguish the consecutive keystrokes. Figure 3 shows the acoustic signal of a push peak and a release peak. We need to detect the start of a keystroke, which is essentially the start of the push peak in a keystroke acoustic signal.

We distinguish between keystrokes and silence using energy levels in time windows. In particular, we calculate windowed discrete Fourier transform of the signal and use the sum of all FFT coefficients as energy. We use a threshold to detect the start of keystrokes. Figure 4 shows an example.

4.1.2 *Features: Cepstrum vs. FFT.* Given the start of each keystroke (i.e. `wav_position`), features of this keystroke are extracted from the audio signal during the period from `wav_position` to `wav_position`$+ \Delta T$. Our experiments compared two different types of features. First we used FFT features with $\Delta T \approx 5$ms, as in [Asonov and Agrawal 2004]. This time period roughly corresponds to the *touch peak* of the keystroke, which is when the finger touches the key. An alternative would be to use the *hit peak*, when the key hits the supporting plate. The hit peak is harder to pinpoint in the signal, so our experiments used the *touch peak*.

As shown in Figure 1, the classification results using FFT features were not satisfactory and we could not achieve the levels reported in [Asonov and Agrawal 2004]. This might be caused by different experimental environment settings, different quality of recording devices, etc.

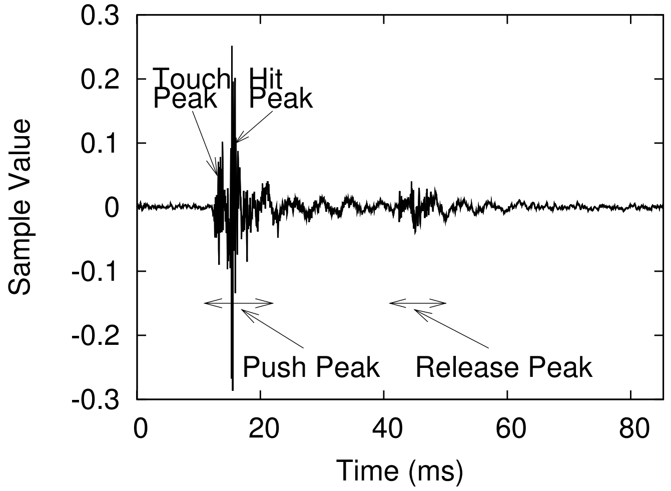

Fig. 3. The audio signal of a keystroke.

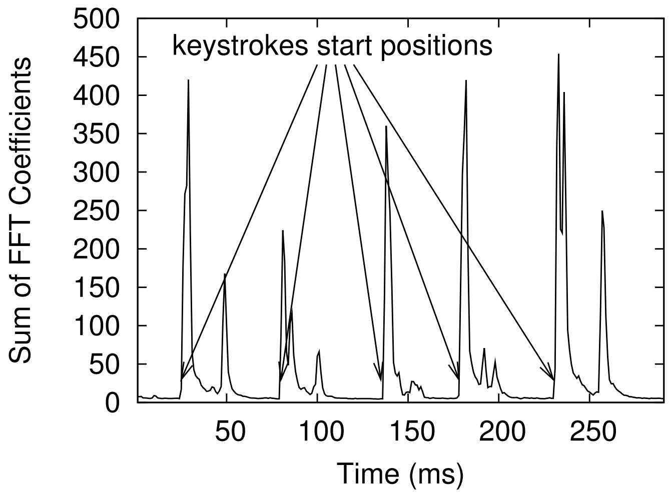

Fig. 4. Energy levels over the duration of 5 keystrokes. (Smaller peaks are release peaks.)

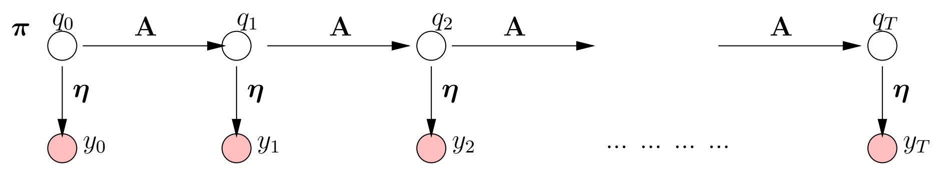

Fig. 5. The Hidden Markov Model for unsupervised key recognition.

Next, we used cepstrum features. Cepstrum features are widely used in speech analysis and recognition [Childers et al. 1977]. Cepstrum features have been empirically verified to be more effective than plain FFT coefficients for voice signals. In particular, we used Mel-Frequency Cepstral Coefficients (MFCCs) [Jurafsky and Martin 2000]. In our experiments, we set the number of channels in the Mel-Scale Filter Bank to 32 and used the first 16 MFCCs computed using 10ms windows, shifting 2.5ms each time. MFCCs of a keystroke were extracted from the period from `wav_position` to `wav_position` $+ \Delta T'$, where $\Delta T' \approx 40$ms which covers the whole push peak. As Figure 1 reports, this yields far better results than from FFT features.

Asonov and Agrawal's observation shows that high frequency acoustic data provides limited value. We ignore data over 12KHz. After feature extraction, each keystroke is represented as a vector of features (FFT coefficients or MFCCs). For details of feature extraction, see Appendix B.

4.2 Unsupervised Single Keystroke Recognition

As discussed above, the unsupervised recognition step recognizes keystrokes using audio recording data only and no training or language data.

The first step is to cluster the feature vectors into K acoustic classes. Possible algorithms to do this include K-means and Expectation-Maximization (EM) on Gaussian mixtures [Bilmes 1997]. Our experiments tested values of K from 40 to 55, and $K = 50$ yielded the best results. We use thirty keys, so K must be equal or larger than 30. A larger K captures more information from the sound samples, but it also makes the system more sensitive to noise. It would be interesting to experiment with using Dirichlet processes that might predict K automatically [Jordan 2005].

The second step is to recover text from these classes. For this we use a Hidden Markov Model (HMM) [Rabiner and Juang 1986]. HMMs are often used to model finite-state stochastic processes. In a Markov chain, the next state depends only on the current state. Examples of processes that are close to Markov chains include sequences of words in a sentence, weather patterns, etc. For processes modeled with HMM, the true *state* of the system is unknown and thus is represented with *hidden* random variables. What is known are *observations* that depend on the state. These are represented with *known* output variables. One common problem of interest in an HMM is the *inference problem*, where the unknown state variables are inferred from a sequence of observations. This is often solved with the Viterbi algorithm [Russell and Norvig 2003]. Another problem is the *parameter estimation problem*, where the parameters of the conditional distribution of the observations are estimated from the sequence of observations. This can be solved with the EM algorithm.

Figure 5 shows the HMM we used. It is represented as a statistical graphical model [Jordan 2005][3]. Circles represent random variables. Shaded circles (y_i) are observations while unshaded circles (q_i) are unknown state variables we wish to infer. Here, q_i is the label of the i-th key in the sequence, and y_i is the class of the keystroke we obtained in the clustering step. The arrows from q_i to q_{i+1} and from q_i to y_i indicate that the latter is conditionally dependent on the former; the value on the arrow is an entry in the probability matrix. So here we have $p(q_{i+1}|q_i) = A_{q_i,q_{i+1}}$, which is the probability of the key q_{i+1} appearing after key q_i. The A matrix is another way of representing plaintext bigram distribution data. The A matrix (called the transition matrix) is determined by the English language and thus is obtained from a large corpus of English text. We also have $p(y_i|q_i) = \eta_{q_i,y_i}$, which is the probability of the key q_i being clustered into acoustic class y_i in the previous step. Our observations (the y_i values) are known. The output matrix η is unknown. We wish to infer the q_i values. Note that one set of values for q_i and η are better than a second set if the likelihood (joint probability) of the whole set of variables, computed by multiplying all conditional probabilities, is larger with the first set than the second set. Ideally, we want a set of values that maximize the likelihood, so we are performing a type of Maximum Likelihood Estimation [Russell and Norvig 2003].

We use the EM algorithm [Bilmes 1997] for parameter estimation. It goes through a number of rounds, alternately improving q_i and η. The output of this step is the η matrix. After that, the Viterbi algorithm [Russell and Norvig 2003] is used to infer q_i, i.e. the best sequence of keys.

EM is a randomized algorithm. Good initial values make the chance of getting satisfactory results better. We found initializing the row in η corresponding to the Space key to an informed guess makes the EM results more stable. This is probably because spaces delimit words and strongly affect the distribution of keys before and after the spaces. This task is performed manually. Space keys are easy to distinguish by ear in the recording because of the key's distinctive sound and frequency of use. We marked several dozen space keys, look at the class that the clustering algorithm assigns to each of them, calculate their estimated probabilities for class membership, and put these into η. This approach yields good results for most of the runs. However, it is not necessary. Even without space keys guessing, running EM with different random initial values will eventually yield a good set of parameters. All other keys used in our study, including punctuation keys are initialized to random values in η. We believe that initialization of η can be completely automated, and hope to explore this idea in the future work.

[3]One might think that a more generalized Hidden Markov Model, such as one that uses Gaussian mixture emissions [Jordan 2005], would give better results. However, the HMM with Gaussian mixture emission has a much larger number of parameters and thus faces the "overfitting" problem. We found a discrete HMM as presented here gave better results.

4.3 Error Correction with a Language Model

As we discussed in Section 3, error correction is a crucial step in improving the results. It is used in unsupervised training, supervised training and also recognition of English text.

4.3.1 *Simple Probabilistic Spelling Correction.* Using a spelling checker is one of the easiest ways to exploit knowledge about the language. We ran spell checks using *Aspell* [Atkinson 2005a] on recognized text and found some improvements. However stock spell checkers are limited in the kinds of spelling errors they can handle, e.g. at most two letters wrong in a word. They are designed to cope well with common errors that human typists make, not the kinds of errors that acoustic emanation classifiers make. It is not surprising that their utility here is limited.

Fortunately, there are patterns in the errors that the acoustic keystroke classifier makes. For example, it may have difficulty with several keys, often confusing one with another. Suppose that we knew the correct plaintext. (This is of course not true, but as we iterate the algorithm, we will predict the correct plaintext with increasing accuracy. Below, we address the case of unsupervised step, where we know no plaintext at all.) Under this assumption, we would have a simple method to exploit these patterns. We act as if this assumption were true, and run the acoustic keystroke classifier on some training data and record all classification results, including errors. With this, we calculate a matrix E (sometimes called the confusion matrix in the machine learning literature),

$$E_{ij} = \hat{p}(y = i | x = j) = \frac{N_{x=j,y=i}}{N_{x=j}} \tag{1}$$

where $\hat{p}(\cdot)$ denotes estimated probability, x is the typed key and y is the recognized key, and $N_{x=j,y=i}$ is the number of times $x = j, y = i$ is observed. Columns of E give the estimated conditional probability distribution of y given x.

Assume that letters are independent of each other and the same is true for words. (This is a false assumption because there is much inter-letter dependence in natural languages, but works well in practice for our experiments.) We compute the conditional probability of the recognized word $\mathbf{Y}$ (the corresponding string returned by the recognizer, not necessarily a correct word) given each dictionary word $\mathbf{X}$.

$$p(\mathbf{Y}|\mathbf{X}) = \prod_{i=1}^{\text{length of }\mathbf{X}} p(\mathbf{Y}_i|\mathbf{X}_i) \approx \prod_i E_{y_i,x_i} \tag{2}$$

In the equation above, $\mathbf{X}_i$ is the i-th character of dictionary word $\mathbf{X}$ and $\mathbf{Y}_i$ is the i-th character of the recognized word. $p(\mathbf{Y}|\mathbf{X})$ represents the probability that the recognition result is Y but the actual user input word is X.

We compute this probability for each dictionary word, which takes only a fraction of a second. The word list we use is SCOWL [Atkinson 2005b] which ranks words by complexity. We use words up to level 10 (higher-level words are more obscure), which covers most commonly used words, giving us 95,997 words in total. By simply selecting the word with the largest posterior probability as our correction result, we correct many errors.

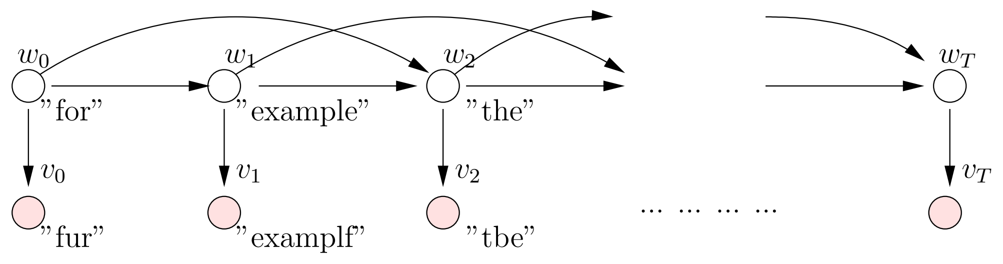

Fig. 6. Trigram language model with spell correction.

Because of the limited amount of training data, there will be many zeroes in E if Equation (1) is used directly, that is, the matrix will be sparse. This is undesirable because the corresponding combination may actually occur in the recognition data. This problem is similar to the zero-occurrence problem in n-gram models [Jurafsky and Martin 2000]. We assign an artificial occurrence count (we use 0.1) to each zero-occurrence event.

In the discussion above we assume the plaintext is known, but we do not even have an approximate idea of the plaintext in the first round of (unsupervised) training. We work around this by letting $E_{ii} = p_0$ where p_0 is a constant (we use 0.5) and distribute the remaining $1 - p_0$ uniformly over all E_{ij} where $j \neq i$. Obviously this gives suboptimal results, but the feedback mechanism corrects this later.

4.3.2 *Adding an n-gram Language Model.* The spelling correction scheme above does not take into account relative word frequency or grammar issues: for example, some words are more common than others, and there are rules in forming phrases and sentences. Spelling correction will happily accept "fur example" as a correct spelling because "fur" is a dictionary word, even though the original phrase is probably "for example".

One way to fix this is to use an n-gram language model that models word frequency and relationship between adjacent words probabilistically [Jurafsky and Martin 2000]. Specifically, we combine trigrams with the spelling correction method above and model a sentence using the graphical model shown in Figure 6. The hidden variables w_t are words in the original sentence. The observations v_t are recognized words. $p(v_t|w_t)$ is calculated using Equation (2) above. Note this is a second-order HMM, because every hidden variable depends on two prior variables. The conditional probability $p(w_t|w_{t-1}, w_{t-2})$ is determined by a trigram model obtained by training on a large corpus of English text.

In this model only the w_i values are unknown. To infer the most likely sentence, we again use the Viterbi algorithm. We use a version of the Viterbi algorithm for second order HMMs, similar to the one in [Thede and Harper 1999]. The complexity of the algorithm is $O(TN^3)$, where T is the length of the sentence and N is the number of possible values for each hidden variable, that is, the number of dictionary words of the appropriate length. To reduce complexity, only the top M candidates from the spelling correction process of each word are considered in the Viterbi algorithm, lowering the cost to $O(TM^3)$. That is, for each recognized word v_t, we select the top M possible hidden variables (w_t) where $p(v_t|w_t)$ are the largest values among all dictionary words. We start from the first word, and each word is chosen from the top M candidate dictionary words. We find the path with the largest:

$$p(v_1|w_1)p(w_2|w_1)p(v_2|w_2)\prod_{t=3}^{T} p(v_t|w_t)p(w_t|w_{t-1}, w_{t-2})$$

We used $M = 20$ in our experiments. Larger M values provide little improvement.

4.4 Supervised Training and Recognition

Supervised training refers to training processes performed with labeled training data. We apply our feedback-based training processes iteratively, using in each iteration characters "recognized" in previous iterations as training samples to improve the accuracy of the acoustic keystroke classifier.

Below, we discuss three different methods of supervised training and recognition we use in our experiments, including the one used in [Asonov and Agrawal 2004]. Like any supervised classification problem, there are two stages:

(1) Training: input feature vectors and corresponding labels (the key pressed) and output a model to be used in recognition;

(2) Recognition: input feature vectors and the trained classification model and output the label of each feature vector (keystroke).

4.4.1 *Method 1: Neural Networks.* The first method is neural networks, also used by Asonov and Agrawal [Asonov and Agrawal 2004]. Specifically, we use probabilistic neural networks, which are arguably the best neural networks available for for classification problems [Wasserman 1993]. We use Matlab's `newpnn()` function, with spread radius parameter as 1.4 (this gives the best results in our experiments).

4.4.2 *Method 2: Linear Classification (Discriminant).* The second method is simple linear (discriminant) classification [Jordan 2005]. This method assumes the data to be Gaussian and finds hyperplanes in the space to divide the classes. We use the `classify()` function from Matlab.

4.4.3 *Method 3: Gaussian Mixtures.* The third method is more sophisticated than linear classification (though it gave worse results in our experiments). Instead of assuming Gaussian distribution of data, it assumes that each class corresponds to a *mixture* of Gaussian distributions [Jordan 2005]. A mixture is a distribution composed of several sub-distributions. For example, a random variable with distribution of a mixture of two Gaussians could have a probability of 0.6 of being in one Gaussian distribution and 0.4 of being in the other Gaussian distribution. This captures the fact that each key may have several slightly different sounds depending on how the typist hit the key.

We also use the EM algorithm to train the Gaussian mixture model. In our experiment, we used mixtures of five Gaussian distributions of diagonal covariance matrices. Mixtures of more Gaussians provide potentially better model accuracy but need more parameters to be trained, requiring more training data and often making EM less stable. We find using five components seems to provide a good

	recording length	number of words	number of keys
Set 1	12m17s	409	2514
Set 2	26m56s	1000	5476
Set 3	21m49s	753	4188
Set 4	23m54s	732	4300

Table I. Statistics of each test set.

tradeoff. Using diagonal covariance matrices reduces the number of parameters. Without this restriction, EM has very little chance of yielding a useful set of parameters.

5. EXPERIMENTS

Our experiments evaluated the attacks. In our first experiment, we worked with four recordings of various lengths of news articles being typed. We use a Logitech Elite cordless keyboard in use for about two years (manufacturer part number: 867223-0100), a $10 generic PC microphone and a Soundblaster Audigy 2 soundcard. The typist was the same for each recording. The keys typed included "a"-"z", comma, period, space and enter. The article was typed entirely in lower case so the shift key was never used. Typists were told to continue typing without using backspace key for error correction. (We discuss these issues in Section 6.) Note that our experiments are preliminary. To validate the applicability of the work in more complicated environment and settings, a more complete set of experiments, in particular with more than a couple typists, would be necessary.

Table I shows the statistics of each test set. Sets 1 and 2 are from quiet environments, while sets 3 and 4 are from noisy environments. Our algorithm for detecting the start of a keystroke sometime fails. We manually corrected the results of the algorithm for sets 1, 2 and 3, requiring ten to twenty minutes of human time per data set. (Sets 1 and 2 needed about 10 corrections; set 3 required about 20 corrections.) For comparison purposes, set 4 (which has about 50 errors in determining the start of keystrokes) was not corrected.

In our second experiment, we recorded keystrokes from three additional models of keyboards (see Section 5.1.2). The same keystroke recognition experiments were run on these recordings and results compared. We used identical texts in this experiments on all these keyboards.

		Set 1		*Set 2*		*Set 3*		*Set 4*	
		words	chars	words	chars	words	chars	words	chars
unsupervised learning	keystrokes	34.72	76.17	38.50	79.60	31.61	72.99	23.22	67.67
	language	74.57	87.19	71.30	87.05	56.57	80.37	51.23	75.07
1st supervised feedback	keystrokes	58.19	89.02	58.20	89.86	51.53	87.37	37.84	82.02
	language	89.73	95.94	88.10	95.64	78.75	92.55	73.22	88.60
2nd supervised feedback	keystrokes	65.28	91.81	62.80	91.07	61.75	90.76	45.36	85.98
	language	90.95	96.46	88.70	95.93	82.74	94.48	78.42	91.49
3rd supervised feedback	keystrokes	66.01	**92.04**	62.70	**91.20**	63.35	**91.21**	48.22	**86.58**
	language	**90.46**	**96.34**	**89.30**	**96.09**	**83.13**	**94.72**	**79.51**	**92.49**

Table II. Text recovery rate at each step. All numbers are percentages. The outputs denoted as "keystroke" are recovery rates before language model correction. The bold face numbers in the "keystroke" row represent recovery rates that could be achieved for random sequences of characters. The outputs denoted as "language" are recovery rates after language model correction. The bold face numbers in the "language" row represent recovery rates that could be achieved for non-random sequences of characters, such as English text.

5.1 English Text Recognition

5.1.1 *A Single Keyboard.* In our experiments, we used linear classification to train the keystroke classifier. Table II shows the result after each step. First, the unsupervised learning step (Figure 2(a)) was run. In this unsupervised step, the HMM model shown in Figure 5 was trained using EM algorithm described above[4]. The output from this step is the recovered text from HMM/Viterbi unsupervised learning, and the text after language model correction. These two are denoted as *keystrokes* and *language* respectively in the table. Then the first round of feedback supervised training produces a new classifier. The iterated corrected text from this classifier (and corresponding text corrected by the language model) are shown in the row marked "1st supervised feedback". We perform three rounds of feedback supervised learning. The bold numbers show our final results. The bold numbers in the "language" row are the final recognition rate we achieve for each test set. The bold numbers in the "keystroke" row are the recognition rates of the keystroke classifier, without using the language model. These are the recognition rates for random or non-English text.

The results show that:

—The language model correction greatly improved the correct recovery rate for words.

—The recovery rates in quiet environments (sets 1 and 2) were slightly better that those in noisy environments (sets 3 and 4). But the difference became smaller after several rounds of feedback.

—Correctness of the keystroke position detection affected the results. The recovery rate in set 3 was better than set 4 because of keystroke location mistakes included in set 4.

—When keystroke positions have been corrected after several rounds of feedback, we achieved an average recovery rate of 87.6% for words and 95.7% for characters.

To understand how different classification methods in the supervised training step affected the results, we reran the same experiment on set 1, using different supervised classification methods. Table III shows our results. The methods in order of quality are is linear classification, then Gaussian mixtures, and then neural networks. Experiments with other data sets gave similar results.

In the experiments above, we used recordings longer than 10 minutes. To discover the minimal amount of training data needed for reasonable results, we took the first data set (i.e. "Set 1" above) and used only the first 4, 5, 7 and 10 minutes of the 12-minute recording for training and recognition. Figure 7 shows the recognition results we get. This figure suggests that at least 5 minutes of recording data are necessary to get good results for this particular recording[5].

[4]Since the EM algorithm is a randomized algorithm, it might sometimes get stuck in local optima. To avoid this, in each of these experiments, we run the same training process eight times and used results from the run with the highest log-likelihood.

[5]The dip in the solid curve probably occuried because of noise during the 2-minute recording window (between minute 5 and minute 7).

		NN		*LC*		*MC*	
		words	chars	words	chars	words	chars
1st supervised feedback	keystrokes	59.17	87.07	58.19	89.02	59.66	87.03
	language	80.20	90.85	89.73	95.94	78.97	90.45
2nd supervised feedback	keystrokes	70.42	90.33	65.28	91.81	66.99	90.25
	language	81.17	91.21	90.95	96.46	80.20	90.73
3rd supervised feedback	keystrokes	71.39	**90.81**	66.01	**92.04**	69.68	**91.57**
	language	**81.42**	**91.93**	**90.46**	**96.34**	**83.86**	**93.60**

Table III. Recognition rates of classification methods in supervised learning. All numbers are percentages. The outputs denoted as "keystroke" are recovery rates before language model correction. The bold face numbers in the "keystroke" row represent recovery rates that could be achieved for random sequences of characters. The outputs denoted as "language" are recovery rates after language model correction. The bold face numbers in the "language" row represent recovery rates that could be achieved for non-random sequences of characters, such as English text. (NN:Neural Network; LC:Linear Classification; MC:Gaussian Mixtures)

		Keyboard 1		*Keyboard 2*		*Keyboard 3*	
		words	chars	words	chars	words	chars
unsupervised learning	keystrokes	30.99	71.67	20.05	62.40	22.77	63.71
	language	61.50	80.04	47.66	73.09	49.21	72.63
1st supervised feedback	keystrokes	44.37	84.16	34.90	76.42	33.51	75.04
	language	73.00	89.57	66.41	85.22	63.61	81.24
2nd supervised feedback	keystrokes	56.34	88.66	54.69	86.94	42.15	81.59
	language	80.28	92.97	76.56	91.78	70.42	86.12
Final result	keystrokes	60.09	**89.85**	61.72	**90.24**	51.05	**86.16**
	language	**82.63**	**93.56**	**82.29**	**94.42**	**74.87**	**89.81**

Table IV. Text recovery rate at each step. With different keyboards. All numbers are percentages. The outputs denoted as "keystroke" are recovery rates before language model correction. The bold face numbers in the "keystroke" row represent recovery rates that could be achieved for random sequences of characters. The outputs denoted as "language" are recovery rates after language model correction. The bold face numbers in the "language" row represent recovery rates that could be achieved for non-random sequences of characters, such as English text.

5.1.2 *Multiple Keyboards.* To verify that our approach applies to different models of keyboards, we performed the keystroke recognition experiment on different keyboards, using linear classification in the supervised training step. The models of the keyboards we used are:

—Keyboard 1: Dell^{TM} Quietkey® PS/2 keyboard, manufacturer part number 2P121, in use for about 6 months.

—Keyboard 2: Dell^{TM} Quietkey® PS/2 keyboard, manufacturer part number 035KKW, in use for more than 5 years.

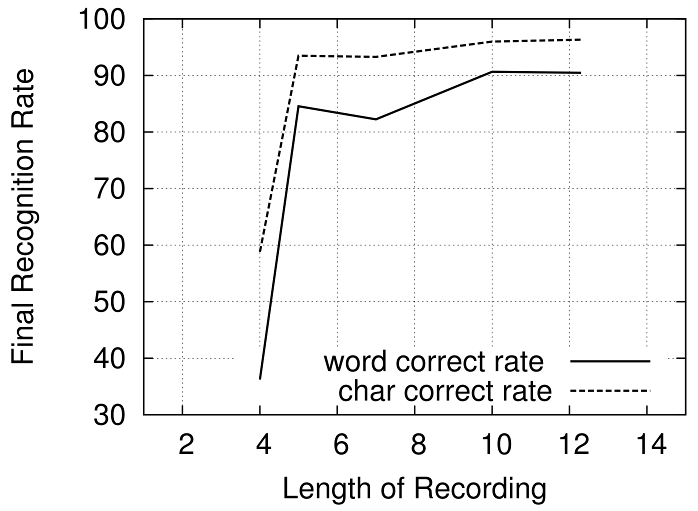

Fig. 7. Length of recording vs. recognition rate.

—Keyboard 3: Dell$^{\text{TM}}$ Wireless keyboard, manufacturer part number W0147, new.

The same document (2273 characters) was typed on all three keyboards and we recorded keystroke sounds. Each recording lasted about 12 minutes. In these recordings, the background machine fan noise was noticeable. While recording from the third keyboard, we got several seconds of unexpected noise from a cellphone nearby. Table IV shows our results. Results in the table show that the first and the second keyboards achieve higher recognition rate than the third one. But in general, all keyboards are vulnerable to the attacks we present in this paper.

5.2 Random Text Recognition and Password Stealing

We used the keystroke classifier trained by set 1 to mount password stealing attacks. All password input recorded in our experiment were randomly generated sequences, not user names or dictionary words. The output of the keystroke classifier for each keystroke is a set of posterior probabilities:

$$p(\text{this keystroke has label } i|\text{observed-sound}), \quad i = 1, 2, \ldots, 30.$$

Given these conditional probabilities, one can calculate probabilities for all sequences of keys being the real password. We sorted these sequences by their probabilities from the largest to the smallest. This produced a candidate list and the attacker can try one-by-one from the top to the bottom. To measure the efficacy of the attack, we used the position of the real password in this list. A user inputed 500 random passwords each of length 5, 8 and 10. Figure 8 shows the cumulative distribution function of the position of the real password. For example, with twenty trials, 90% of 5-character passwords, 77% of 8-character passwords and 69% of 10-character passwords are recovered. As Figure 8 also shows, after seventy-five trials, we can recover 80% of 10-character passwords.

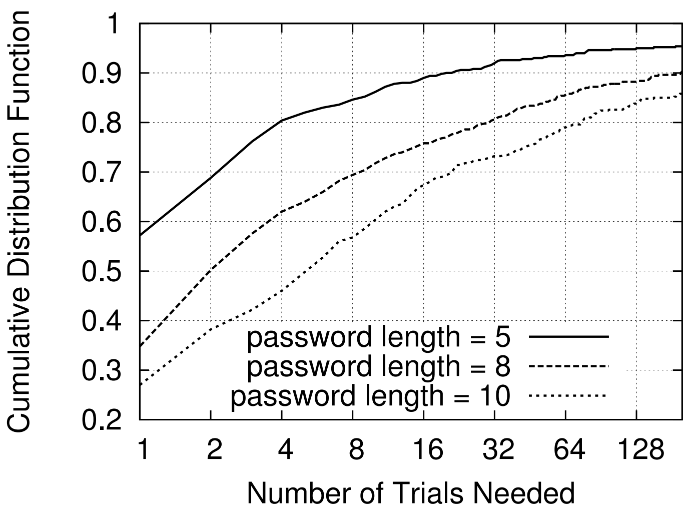

Fig. 8. Password stealing: distribution of the number of trials required by the attacker.

6. DISCUSSION

6.1 Timing Information

We discuss above how to use acoustic information of keystrokes to recover typed keys. Our experiments show high keystroke recovery rates using only acoustic information. As mentioned above, Song, Wagner and Tian point out that the time between consecutive keys also carries information about typed keys [Song et al. 2001]. It may be possible to further improve the recovery rate with timing information. Here we give one way to combine timing features with acoustic features, however our results show only a modest improvement in recovery rate.

The time between a pair of consecutive keys is related to many factors, such as the location of the two keys on the keyboard, typing style, whether the keys are typed by alternating hands or the same hands, whether the keys are typed by different fingers or the same finger, etc. We recorded a typist at normal pace, without intentional stops between keys. Figure 9 shows the distribution of time between "a" and subsequent keys is significantly different from "h" and subsequent keys. The key "a" is located near the border of a keyboard and touch typists use the small finger of the left hand to type it; while the key "h" is located in the middle of a keyboard and touch typists use the index finger of the right hand to type it. Figure 9 suggests the time between a key and a subsequent key carries information about the location of the key on the keyboard, that is, information related to the label of the key.

In the discussion above we represent acoustic information as a vector of features. If we assume the length of the time interval between a key and its next key carries information (as shown in Figure 9), we can add time as an additional dimension in the feature vector. We can then apply new feature vectors with time as one of the dimensions in our supervised training step. We experimented with the sets 1, 2, and 3, using training approaches as above. Table V shows that initial and final

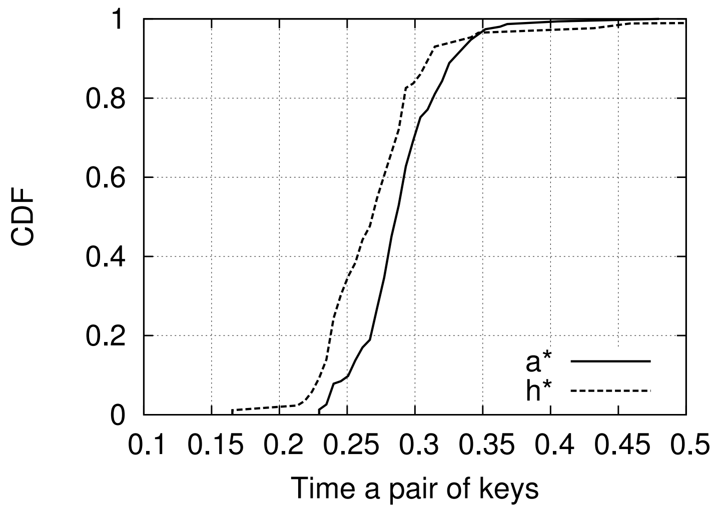

Fig. 9. Cumulative Distribution Function (CDF) of time: a pair of keys starting with "a"("a*") vs. a pair of keys starting with "h"("h*").

recognition rates in supervised training.

			without time		with time	
			word	char	word	char
set 1	initial	keystroke	58.19	89.02	57.95	88.94
		language	89.73	95.94	89.00	95.31
	final	keystroke	66.01	92.04	65.77	92.12
		language	90.46	96.34	**91.20**	**96.50**
set 2	initial	keystroke	58.20	89.86	58.20	89.83
		language	88.10	95.64	87.7	95.28
	final	keystroke	62.70	91.20	62.70	91.09
		language	89.30	96.09	89.2	96.00
set 3	initial	keystroke	51.53	87.37	51.39	87.32
		language	78.75	92.55	77.56	92.09
	final	keystroke	63.35	91.21	62.55	91.07
		language	83.13	94.72	82.20	94.44

Table V. Recognition rate in supervised training: with timing information vs. without timing information.

The recognition rates in Table V suggest that time between consecutive keys does not substantially improve the supervised learning in the feedback based training. The results are not as good as the results reported by Song, Wagner and Tian. Reasons for this discrepancy may include:

—Song et al. used the length of the time interval between consecutive keys in a very short phase, such as a password. The pace of typing when a user types his password is probably more consistent than the pace of typing when a user types an article. In our test sets, the typist sometimes stopped in the middle of typing an article. Also typing speed for some words are much faster than others even if those words share common pairs of characters. The newly introduced timing information ("signal") comes along with random variations ("noise") above. When we add the timing information with acoustic information, the training methods do not receive a sufficient number of samples with consistent timing information to improve recovery rates.

—Acoustic information alone yields a high recognition rate. The acoustic information has a higher "signal-to-noise" ratio compared to timing information. Moreover, spelling and grammar correction makes the effects of timing information less visible too.

Finding good ways to combine inter-keystroke time interval information with acoustic key recovery merits further research.

6.2 Why Keys Sound Different

We are interested in finding out why keystrokes of different keys sound different. There are at least two contributing factors:

—*Keyboard layout.* Keys are located at different locations on the support plate of a keyboard. Just as striking a drum at different locations yields different sounds, the keys on a keyplate yield different sounds.

—*Typing patterns.* The sound of keystroke is related to how the key is typed; for example, the direction that a key is hit.

To verify the hypotheses above, we performed an experiment.

In the experiment, each key was repeatedly struck 50 times. The sound samples of 50 hits for each key were used to train acoustic classifiers using different classification methods (i.e. linear classification, neural networks and mixture of Gaussians). Then, the acoustic classifiers trained in this way were used to recognize the training set and two different sets of new sound samples. The first test set was composed of sound samples from 30 repeated hits of each key. The second test set is composed of sound samples from a typist typing an article. Table VI shows our recognition rates.

	repeat50 (training)	*repeat30*	*article*
Linear Classification	*95.67%*	88.05%	53.49%
Neural Network	*100%*	81.84%	51.21%
Mixture of Gaussians	*98.87%*	81.15%	47.44%

Table VI. Recognition rate of repeat key hits and article input using classifier trained by repeat key hits.

When a key was repeatedly struck, all keystrokes are made with the similar strength by a single finger and using almost the same gesture. There is no difference in typing style between different keys. Table VI shows that the recognition rates of test samples from keys typed repeatedly are over 80%, which suggests that the sound differences may come from the physical properties of a keyboard: location of keys, physical difference between keys, etc. This observation supports our first assumption that keyboard layout contributes to different sound from keys.

If the keyboard layout was the only reason for different sound of keys, the classifier trained by sound of repeatedly typing the same key should also work for normal typing. However, this model was not very effective in classifying normal English text. One reason for this is that typing normal English text uses a variety of paces, gestures, and key press strengths. Repeated typing only teaches the acoustic classifiers a portion of sounds that a key could make. This experiment suggests that keys sound different because of both the keyboard layout and typing style such as paces, gestures and hitting strengths, etc.

6.3 Special Keys

The current attack does not take into account special keys such as the Shift key, the Control key, the Backspace key and the Caps Lock key. There are two issues here. One is whether keystrokes of special keys are separable from other keystrokes at signal processing time. Our preliminary experiments suggest this is possible; push peaks of keystrokes are easily separable in the recordings we looked at. The other issue is how modifier keys such as the Shift key fit into spelling correction scheme. We hypothesize ad hoc solutions such as replacing the Shift key or the Caps Lock key with spaces will work. The Shift key often appears before the first letter of a word. If it is recognized incorrectly, the following word will be one letter longer. In spelling and grammar correction, we can take this into account by not only considering words of the same lengths, but also those with one fewer letter. For example, if we get "atje" after initial recognition, the word "the" will also be considered as a candidate word for correction because we might misrecognize the Shift key as "a". These keys can be much more reliably recognized by training a classifier specifically for the Shift key and the Caps Lock key. Note that we do not need to distinguish between uppercase and lowercase in the recovered text, so it is not necessary to detect when the Shift key is released.

The Backspace key is also important. The ideal solution would be to figure out what the final text is after applying the backspaces. But that probably will complicate the error correction algorithms. So one could just recognize these keys and leave the "word" before and after out of error-correction because they are probably not full words. An interesting fact about the Backspace key is that this key is often hit repeatedly: a user often wants to delete a whole word or a whole sentence. Here, a bit of human aid could be useful because backspaces are relatively easy to detect by ear based on sound and context, although it is harder than spaces. It is not difficult for human ears to detect repeated keystrokes of the Backspace key. Since the sound of the Backspace key is very different from others, in the acoustic clustering step, they will normally clustered with the same label. It is possible to write a program to automatically select sound samples of consecutive keys which are clustered in a common label. A variety of techniques could be used to decide whether these are consecutive Backspaces. After sound samples of the Backspace keys are collected, we train a specific acoustic classifier for the Backspace keys as well.

6.4 Attack Improvements

This section discusses improvements to our attacks.

—One challenge we met in our work was marking keystroke starting points in the sound signal. This is not trivial because the sound varies in energy level, timing, frequency distribution, etc., depending on the typist and recording environment. We use energy level and timing constraints between consecutive keys to mark the starting positions of keystrokes. Detection rules are manually created based on past experiences. Our detection program based on this approach has difficulty in marking keystroke positions in recordings from fast typists. However, there is additional information we can use: namely frequency, which appears to vary from the push peak to the release peak. We hope to explore this in future work. A robust and consistent keystroke position detection algorithm may also improve the recovery rate of typed characters.

—Currently, we assume the Space key, Enter key and punctuation keys are detected correctly and use them to divide characters into words. We use candidate words of the same length as the "words" separated in this way. In future work, we will explore better ways to choose candidate words for correction, with the goal of high quality correction even when there are mistakes in separating words.

—An alternative method for feedback training is Hierarchical Hidden Markov Models (HHMMs) [Fine et al. 1998]. In a HHMM, HMMs of multiple levels (grammar level and spelling level in this case) are built into a single model. Algorithms to maximize global joint probability may improve the effectiveness of the feedback training procedure. This approach merits further investigation.

—Our experiments tested on FFT features and cepstrum features. However, there are other types of features for representing sound signals. For each type of feature, there are multiple parameters to control the extracted information. Currently, we used ad hoc methods to select these parameters. An entropy based metric defined specifically for measuring acoustic features may provide better, more systematic way to compare features and parameters. This metric may also allow us to compare information leaked by individual keys. Given current PC keyboard layouts, is the leaking uniform among keys, or should we pay more attention to specific keys? Is it possible to know which typing styles leak more information and whether different typists leak different amounts of information?

—In a controlled environment where we can record isolated typing sounds, the recovery rate is now high. However, in most realistic situations, environmental background noise is an issue. In many work spaces, we have multiple users simultaneously typing. Isolating the sound of a single typist is difficult. We propose to test recording with multiple microphones, including arrays of directional microphones. We could get the sound signal of multiple channels in this way. Similarly, we have shown that the recognition rate is lower in noisy environments. Attacks will be less successful when the user is playing music or talking to others while typing. However, we may be able to use signal processing techniques (especially in multichannel recordings) to isolate the sound of a single typist.

—We hope to explore a variety of recording devices including parabolic microphones, laser microphones, telephone receiver microphones, acoustic chat connections such as Skype, etc.

—In future work, it is particularly interesting to try to detect keystrokes typed in a particular application, such as a visual editor (e.g. emacs) or a software development environment (e.g. Eclipse). Examining text typed in these environment presents challenges because more keys maybe used and special keys maybe used more often. Furthermore, the bigram or transition matrix will be different. Nonetheless we believe that our techniques may be applicable to detecting keystrokes of users in these applications and indeed can even cover input as different as other small alphabet languages, such as Russian or Arabic, large alphabet languages, such as Chinese or Japanese, and even programming languages.

6.5 Defenses

Since our attack is based on acoustic signal through passively eavesdropping, it is more difficult to detect this type of attacks than active attacks where attackers actively interact with victims. Here are some preliminary areas for potential defenses:

—Reduce the possibility of leaking acoustic signals. Sound proving may help, but given the effectiveness of modern parabolic and laser microphones, the standards are very high.

—Quieter keyboards as suggested by Asonov and Agrawal may reduce vulnerability. However, the two so-called “quiet” keyboards we used in our experiments proved ineffective against the attack. Asonov and Agrawal also suggest that keyboard makers could produce keyboards having keys that sound so similar that they are not easily distinguishable. They claim that one reason keys sound different today is that the plate underneath the keys makes different sounds when hit at different places. If this is true, using a more uniform plate may alleviate the attack. However, it is not clear whether these kinds of keyboards are commercially viable. Also, there is the possibility that more subtle differences between keys can still be captured by an attacker. Further, keyboards may develop distinct keystroke sounds after months of use.

—Another approach is reduce the quality of acoustic signal that could be acquired by attackers. We could add masking noise while typing. However, we are not sure that masking noises might not be easily separated out. As we discussed above, an array of directional microphones may be able to record and distinguish sound into multiple channels according to the locations of the sound sources. This defense could also be ineffective when attackers are able to collect more data. Reducing the annoyance of masking is also an issue. Perhaps a short window of noise could be added at every predicted push peak. This may be more acceptable to users than continuous masking noise. Alternatively, perhaps we could randomly insert noise windows which sound like push peaks of keystrokes.

—The practice of relying only on typed passwords or even long passphrases should be reexamined. One alternative is two-factor authentication that combines passwords or pass-phrases with smart cards, one-time-password tokens, biometric authentication and etc. However two-factor authentication does not solve all our problems. Typed text other than passwords is also valuable to attackers.

7. CONCLUSION

Our new attack on keyboard emanations needs only acoustic recording of typing using a keyboard and recovers the typed content. Compared to previous work that requires clear-text labeled training data, our attack is more general and serious. More important, the techniques we use to exploit inherent statistical constraints in the input and to perform feedback training can be applied to other emanations with similar properties.

8. ACKNOWLEDGMENTS

This work is funded by US National Science Foundation contracts EIA-01225989, IIS-0205647, CNS-0325247 and CCS-0424422. This paper does not necessarily reflect the views of the US government or any funding sponsor.

We would like to thank the anonymous reviewers for their comments. We also want to thank Chris Karlof, Michael I. Jordan, Naveen Sastry, Umesh Shankar, Zile Wei, and Hao Zhang, and especially Marco Barreno.

REFERENCES

Asonov, D. and Agrawal, R. 2004. “Keyboard Acoustic Emanations”. In *Proceedings of the IEEE Symposium on Security and Privacy.* 3–11.

Atkinson, K. 2005a. *GNU Aspell.* `http://aspell.sourceforge.net/`.

Atkinson, K. 2005b. *Spell Checker Oriented Word Lists.* `http://wordlist.sourceforge.net/`.

Bar-El, H. 2003. Introduction to Side Channel Attacks. `http://www.hbarel.com/Misc/side_channel_attacks.html`.

Bilmes, J. A. 1997. *A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models.* Technical Report ICSI-TR-97-021, International Computer Science Institute, Berkeley, California.

Briol, R. 1991. “Emanation: How to Keep Your Data Confidential”. In *Proceedings of Symposium on Electromagnetic Security For Information Protection.* 225–234.

Childers, D. G., Skinner, D. P., and Kemerait, R. C. 1977. “The Cepstrum: A Guide to Processing”. In *Proceedings of the IEEE, Vol. 65, No. 10.* 1428–1443.

Fine, S., Singer, Y., and Tishby, N. 1998. “The Hierarchical Hidden Markov Model: Analysis and Applications”. *Machine Learning 32,* 1, 41–62.

Jordan, M. I. 2005. *An Introduction to Probabilistic Graphical Models.* In preparation.

Jurafsky, D. and Martin, J. H. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall.

Kuhn, M. G. 2002. “Optical Time-Domain Eavesdropping Risks of CRT Displays”. In *Proceedings of the IEEE Symposium on Security and Privacy.* 3–18.

Kuhn, M. G. 2003. “Compromising Emanations: Eavesdropping Risks of of Computer Displays”. Technical Report UCAM-CL-TR-577, Computer Laboratory, University of Cambridge.

Rabiner, L. R. and Juang, H. 1986. “An Introduction to Hidden Markov Models”. In *IEEE Acoustics, Speech, and Signal Processing Magazine, Vol. 3.* 4–16.

Russell, S. and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach*, 2nd ed. Prentice Hall.

Shamir, A. and Tromer, E. 2004. *Acoustic Cryptanalysis.* `http://www.wisdom.weizmann.ac.il/~tromer/acoustic/`.

Song, D., Wagner, D., and Tian, X. 2001. Timing analysis of keystrokes and timing attacks on ssh. In *Proceeding of the 10th USENIX Security Symposium.* 337–352.

Thede, S. M. and Harper, M. P. 1999. “A Second-order Hidden Markov Model for Part-of-speech Tagging”. In *Proceedings of the 37th conference on Association for Computational Linguistics.* 175–182.

Wasserman, P. D. 1993. *Advanced Methods in Neural Computing.* Wiley.

A. RECOVERED TEXT EXAMPLES

Text recognized by the HMM classifier, with cepstrum features (underlined words are wrong),

```
the big money fight has drawn the shoporo od dosens of companies
in the entertainment industry as well as attorneys gnnerals on
states, who fear the fild shading softwate will encourage illegal
acyivitt, srem the grosth of small arrists and lead to lost cobs
and dimished sales tas revenue.
```

Text after spelling correction using trigram decoding,

```
the big money fight has drawn the support of dozens of companies
in the entertainment industry as well as attorneys generals in
states, who fear the film sharing software will encourage illegal
activity, stem the growth of small artists and lead to lost jobs
and finished sales tax revenue.
```

Original text. Notice that it actually contains two typographical errors, one of which is fixed by our spelling corrector.

```
the big money fight has drawn the support of dozens of companies
in the entertainment industry as well as attorneys gnnerals in
states, who fear the file sharing software will encourage illegal
activity, stem the growth of small artists and lead to lost jobs
and dimished sales tax revenue.
```

B. DETAILS OF FEATURE EXTRACTION IMPLEMENTATION

The main difference between the duration of keystrokes and the silent periods between keystrokes is the level of energy in a certain range of frequencies. The "silent" periods between keystrokes might also have non-negligible energy because of other noises. The major part of energy is in different frequency range than those from keystrokes. Our experiments show that the energy of keystroke durations is mainly in the frequencies between 400Hz and 12KHz.

To extract the start of each keystroke, we:

(1) Compute the windowed discrete-time Fourier transform of a signal using a sliding window (e.g. Matlab `specgram`()) with the magnitude of outputs as the spectrogram;

(2) Sum over the spectrogram in the range $[0.4, 12]$ KHz to get a aggregate curve;

(3) Set a threshold and find the start of each peak in the curve as the start of a keystroke (see Figure 4).

Note that the positions of starts of keystrokes detected from the curve in Figure 4 is the index of window number (`win_num`), which are converted back to the original location (`wav_position`) in the audio stream by:

$$\texttt{wav\_position} = (\texttt{win\_num} - 2) * \texttt{win\_shift} + \texttt{win\_length}$$

The raw features might have high dimensionality. Possible algorithms for dimensionality reduction are Factor Analysis (FA) or the simpler Principal Component Analysis (PCA) [Jordan 2005]. Although some of our preliminary experiments use PCA, our final experiments do not use it. The FFT and cepstrum features we extract are not of very high dimension (typically the number of dimension is between 60 and 80), so we do not need to apply dimension reduction.