Uncovering Spoken Phrases in Encrypted Voice over IP Conversations

CHARLES V. WRIGHT
MIT Lincoln Laboratory
LUCAS BALLARD
Google Inc.
SCOTT E. COULL, FABIAN MONROSE
University of North Carolina, Chapel Hill
and
GERALD M. MASSON
Johns Hopkins University

Although Voice over IP (VoIP) is rapidly being adopted, its security implications are not yet fully understood. Since VoIP calls may traverse untrusted networks, packets should be encrypted to ensure confidentiality. However, we show that it is possible to *identify the phrases spoken within encrypted VoIP calls* when the audio is encoded using variable bit rate codecs. To do so, we train a hidden Markov model using only knowledge of the phonetic pronunciations of words, such as those provided by a dictionary, and search packet sequences for instances of specified phrases. Our approach does not require examples of the speaker's voice, or even example recordings of the words that make up the target phrase. We evaluate our techniques on a standard speech recognition corpus containing over 2,000 phonetically rich phrases spoken by 630 distinct speakers from across the continental United States. Our results indicate that we can identify phrases within encrypted calls with an average accuracy of 50%, and with accuracy greater than 90% for some phrases. Clearly, such an attack calls into question the efficacy of current VoIP encryption standards. In addition, we examine the impact of various features of the underlying audio on our performance and discuss methods for mitigation.

Categories and Subject Descriptors: K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms: Security

Additional Key Words and Phrases: Voice over IP, Traffic Analysis, Network Security

1. INTRODUCTION

Over the past few years, Voice over IP (VoIP) has become an attractive alternative to more traditional forms of telephony. Naturally, with its increasing popularity in

This work was supported in part by the U.S. Department of Homeland Security Science & Technology Directorate under Contract No. FA8750-08-2-0147 and by NSF grant CNS-0546350.

daily communications, researchers are continually exploring ways to improve both the efficiency and security of this new communication medium. Unfortunately, while it has been well understood for some time now that VoIP packets must be encrypted to ensure confidentiality [Provos 2004], it has been shown that simply encrypting packets may not be sufficient from a privacy standpoint. For instance, we recently showed that it is possible to determine the spoken language of the encrypted conversation when VoIP packets are first compressed with variable bit rate (VBR) encoding schemes to save bandwidth, and then encrypted with a length preserving stream cipher to ensure confidentiality [Wright et al. 2007].

As surprising as these findings may be, some might argue that learning the language of the speaker (e.g., Arabic) only affects privacy in a marginal way. If both endpoints of a VoIP call are known (for example, Mexico City and Madrid), then one might correctly conclude that the language of the conversation is Spanish, without performing any analysis of the traffic. In this paper, we show that the information leaked from the combination of using VBR and length preserving encryption is indeed much more serious. Specifically, we demonstrate that it is possible to spot arbitrary phrases of interest within the encrypted conversation. Our techniques achieve far greater precision than one would expect, thereby calling the effectiveness of the encryption scheme into question.

At a high level, the success of our technique stems from exploiting the correlation between the most basic building blocks of speech—namely, *phonemes*—and the length of the packets that a VoIP codec outputs when presented with these phonemes. Intuitively, to search for a word or phrase, we first build a model by decomposing the target phrase into its most likely constituent phonemes, and then further decomposing those phonemes into the most likely packet lengths. Next, given a series of packet lengths that correspond to an encrypted VoIP conversation, we simply examine the output stream for a subsequence of packet lengths that match our model. Of course, speech naturally varies for any number of reasons, and so two instances of the same word will not necessarily be encoded the same way. Therefore, to overcome this, we make use of *profile hidden Markov models* [Durbin et al. 1999] to build a speaker-independent model of the speech we are interested in finding. Using these models we are then able to determine when a series of packets is similar to what we would expect given a set of phonemes.

As we show later, the approach we explore is accurate, even in the face of very little information. We assume that an attacker only has access to (*1*) the ciphertext she wishes to search, (*2*) knowledge of the spoken language of the conversation (e.g., using our earlier techniques [Wright et al. 2007] she may know this is a Spanish conversation), and (*3*) statistics defining what phonemes are mapped to what packet lengths by the VoIP codec. We argue that even the last assumption is realistic, as this information can be readily gathered by an adversary who can use the codec as a "black box" to compress prerecorded speech. For example, in the case of English, there are relatively few phonemes and therefore it is plausible to assume that the attacker can find sufficiently many instances of each phoneme to generate realistic models. She can then use these phonemes to construct models even for words she has not seen before.

Our results show that an eavesdropper who has access to neither the speaker's

voice nor even a single utterance of the target phrase, can identify instances of the phrase with average accuracy greater than 50%. In some cases, accuracy can exceed 90%. Clearly, any system that is susceptible to such attacks provides only a false sense of security to its users. These results are particularly unnerving when one considers that many widely available VoIP products, including the popular Skype software, makes use of VBR [Zimmerman 2008]. In the following sections, we evaluate the effectiveness of our attack under a variety of conditions to understand its real-world implications. Additionally, we explore methods to mitigate the information leaked from encrypted VoIP. Beyond the technical details presented in our earlier work [Wright et al. 2008], we also investigate the performance of our method when allowing for partial matches of a phrase, and provide a detailed examination of the features that impact our ability to spot phrases.

The remainder of the paper is organized as follows. In Section 2 we develop intuition for why VBR-encoded speech is vulnerable to phrase spotting and show some exploratory data analysis in support of this intuition. The relevant background for understanding profile HMMs and the workings of our search algorithm are given in Section 3. Section 4 presents our experimental methodology and results. In Section 5 we investigate the factors that determine the effectiveness of our algorithm on different phrases, and in Section 6 we evaluate techniques for thwarting our attack. Finally, we review related work in Section 7 and conclude in Section 8.

2. INTUITION AND EXPLORATORY DATA ANALYSIS

In what follows, we briefly review the principles of speech coding and speech recognition that are most relevant to Voice over IP and to our attack. In VoIP, connection setup and the transmission of voice data are typically performed using separate connections. The control channel operates using a standard application-layer protocol like the Session Initiation Protocol (SIP) [Rosenberg et al. 2002], the Extensible Messaging and Presence Protocol (XMPP) [Saint-Andre 2004], or an application-specific control channel like Skype [Skype 2009]. The voice data is typically transmitted as a Real-time Transport protocol (RTP) [Schulzrinne et al. 1996] stream over UDP, which carries a version of the audio that has been compressed using a special-purpose speech codec such as GSM [ETSI/GSM 1991], G.728 [Dimolitsas et al. 1993], or several others.

Generally speaking, the codec takes as input the audio stream from the user, which is typically sampled at either 8,000 or 16,000 samples per second (Hz). At some fixed interval, the codec takes the n most recent samples from the input, and compresses them into a packet for efficient transmission across the network. To achieve the low latency required for real-time performance, the length of the interval between packets is usually fixed between 10 and 50ms, with 20ms being the common case. Thus, for a 16kHz audio source, we have $n = 320$ samples per packet, or 160 samples per packet for the 8kHz case.

Many common voice codecs are based on a technique called code-excited linear prediction (CELP) [Schroeder and Atal 1985], which is shown in Figure 1. For each packet, a CELP encoder simply performs a brute-force search over the entries in a codebook of audio vectors to output the one that most closely reproduces the original audio. The quality of the compressed sound is therefore determined by the

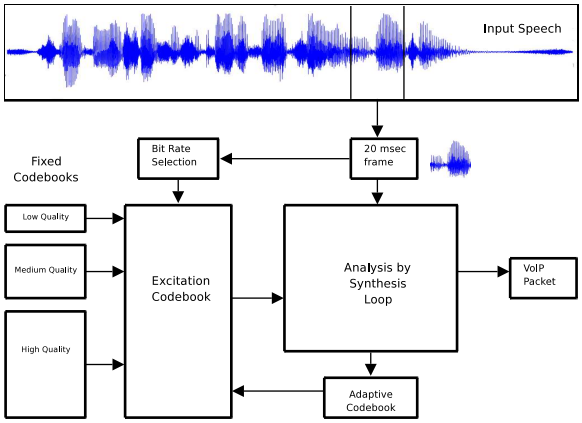

Fig. 1. Basic CELP encoder

number of entries in the codebook. The index of the best-fitting codebook entry, together with the linear predictive coefficients and the gain, make up the payload of a CELP packet. The larger codebooks used for higher-quality encodings require more bits to index, which results in higher bit rates and therefore larger packets.

In some CELP variants, such as QCELP [Gardner et al. 1993], Speex's [Valin and Montgomery 2006] variable bit rate mode, or the approach advocated by Zhang et al. [Zhang et al. 1997], the encoder adaptively chooses the bit rate for each packet in order to achieve a good balance of audio quality and network bandwidth. This approach is appealing because the decrease in data volume may be substantial, with little or no loss in quality. For instance, in a two-way call, each participant is idle roughly 63% of the time [Chu 2003], and therefore much of the conversation can be compressed with low bit rates. Unfortunately, this approach can also cause substantial information leakage within encrypted VoIP calls because, in the standard specification for Secure RTP (SRTP) [Baugher et al. 2004], the cryptographic layer does not pad or otherwise alter the size of the original RTP payload.

Intuitively, the sizes of CELP packets leak information because the choice of bit rate is largely based on the audio encoded in the packet's payload. For example, the variable bit-rate Speex codec encodes *vowel* sounds at higher bit rates than *fricative* sounds like "*f*" or "*s*". In phonetic models of speech, sounds are broken down into several different categories, including the aforementioned vowels and fricatives, as well as *stops* like "*b*" or "*d*", and *affricatives* like "*ch*". Each of these canonical sounds is called a *phoneme*, and the pronunciation for each word in the language can then be given as a sequence of phonemes. While there is no consensus on the exact number of phonemes in the English language, most in the speech community put the number between 40 and 60. For a listing of one commonly-used set of 60 English phonemes, with examples of each, see Appendix A.

To demonstrate the relationship between bit rate and phonemes, we encoded several recordings from the TIMIT [Garofolo et al. 1993] corpus of phonetically-rich English speech using Speex in wideband (i.e., 16 kHz sampling) variable bit rate mode, and observed the bit rate used to encode each phoneme. The probabilities

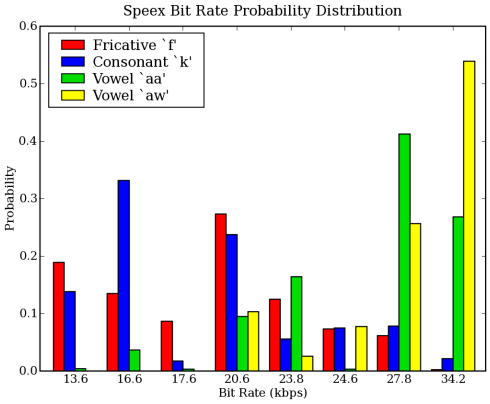

Fig. 2. Distribution of bit rates used to encode four phonemes with Speex

for 8 of the 21 possible bit rates are shown for a handful of phonemes in Figure 2. As expected, we see that the two vowel sounds, "*aa*" and "*aw*", are typically encoded at significantly higher bit rates than the fricative "*f*" or the consonant "*k*". Moreover, large differences in the frequencies of certain bit rates (e.g., 16.6, 27.8, and 34.2 kbps), can be used to distinguish *aa* from *aw* and *f* from *k*.

It is these differences in bit rate for the phonemes that make recognizing words and phrases in encrypted traffic possible. To further illustrate the patterns that occur in the stream of packet sizes when a certain word is spoken, we examined the sequences of packets generated by encoding several utterances of the words "artificial" and "intelligence" from the TIMIT corpus [Garofolo et al. 1993]. We represent the packets for each word visually in Figures 3 and 4 as *heatmaps*. We show the encoder's bit rate on the y-axis and position in the sequence on the x-axis. Starting with a plain white background, we darken the cell at position (x, y) each time we observe a packet encoded at bit rate y and position x for the given word.

In both graphs, we see several dark gray or black grid cells where the same packet size is consistently produced across different utterances of the word, and in fact, these dark spots are closely related to the phonemes in the two words. In Figure 3, the bit rate in the 2^{nd}–5^{th} packets, which is associated with the "*a*" in artificial, is usually quite high (35.8kbps), as we would expect for a vowel sound. Then, in packets 12–14 and 20–22, we see much lower bit rates for the fricative "*f*" and affricative "*sh*". Similar trends are visible in Figure 4; for example, the "*t*" sound maps consistently to 24.6 kbps in both words.

3. SPOTTING PHRASES WITH PROFILE HMMS

Our goal in this work is to recognize spoken phrases in encrypted VoIP conversations using only minimal knowledge of what the actual audio content of the phrase should sound like. In fact, the techniques we develop here do not require knowledge of the identity of the speaker, or any examples of the audio produced by speaking the target word or phrase. For ease of exposition, we begin the discussion of our

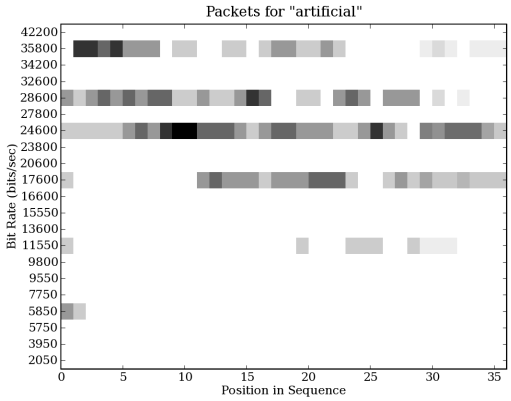

Fig. 3. Bit rates for "artificial"

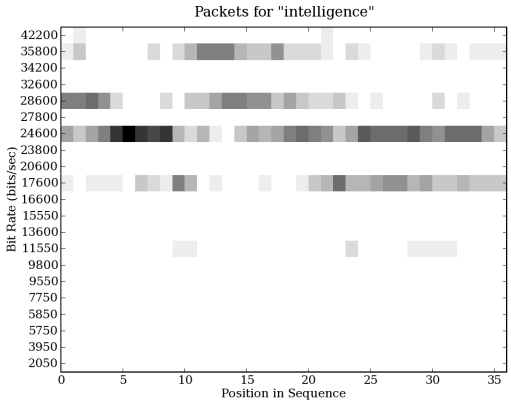

Fig. 4. Bit rates for "intelligence"

machine learning techniques by first addressing a much easier scenario, wherein the attacker does have access to several recordings of the target phrase being spoken, though not necessarily by the target speaker. Later, we show how these techniques can be adapted to handle the more challenging case where the attacker may have never heard the word or phrase she wishes to detect.

3.1 Recognizing a phrase by training from several examples

If we assume that the same sequence of packet sizes is produced each time a given word is spoken, then the problem of identifying instances of that word can be reduced to a substring matching problem. However, human speech is known to exhibit a high degree of variability, and the adaptive compression performed by the codec may contribute additional variance to the resulting stream of packet sizes. To handle this variation, we can instead apply matching algorithms from the speech recognition and bioinformatics communities. In both of these areas, techniques

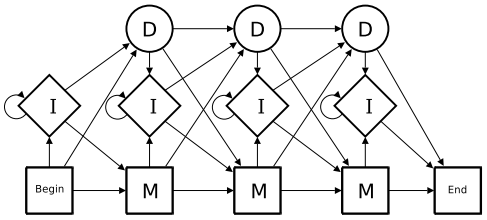

Fig. 5. An example profile HMM

based on hidden Markov models [Rabiner 1989] have proven to be extremely useful [Wilpon et al. 1990; Durbin et al. 1999]—especially when the training data itself may exhibit high variability.

In particular, the common bioinformatics problem of searching a protein database for fragments of known protein families is similar in many ways to searching a stream of packet sizes for instances of a word or phrase. Proteins are made up of twenty different amino acids, while the Speex codec in wideband mode produces twenty one distinct packet sizes. There may be significant variation between proteins in the same family or between different utterances of the same phrase. Therefore, in this paper, we adapt *profile hidden Markov model* techniques [Eddy 1995], which were originally developed for performing multiple sequence alignment of protein families and searching protein databases [Krogh et al. 1994], to the task of finding words and phrases in encrypted VoIP traffic.

The general outline of our strategy is as follows: *(1)* build a profile HMM for the target phrase, *(2)* transform the profile HMM into a model suitable for performing searches on packet sequences, and *(3)* apply Viterbi decoding [Viterbi 1967] on the stream of packets to find subsequences of packets that match the profile. We elaborate on each of these steps below.

3.1.1 *Building a Profile HMM.* A profile HMM [Durbin et al. 1999], shown in Figure 5, consists of three interconnected chains of states, which describe the expected packet lengths at each position in the sequence of encrypted VoIP packets for a given phrase. The *Match* states, shown in Figure 5 as squares, represent the expected distribution of packet sizes at each position in the sequence. *Insert* states, shown as diamonds, and *Delete* states, shown as circles, allow for variations from the typical sequence. The Insert states emit packets according to a uniform distribution or some other distribution that represents the overall frequencies of packet sizes in VoIP streams, and thus they allow for additional packets to be "inserted" in the expected sequence. Delete states are *silent*, meaning that they simply transition to the next state without emitting any packets; doing so allows for packets that are normally present to be omitted from the sequence. Initially, the Match states' emission probabilities are set to a uniform distribution over packet sizes, and the transition probabilities in the model are set in such a way that the Match states are the most likely state in each position.

Given an initial model and a set of example sequences of packets for the target phrase, there is a well-known Expectation-Maximization [Dempster et al. 1977] algorithm due to Baum and Welch [Baum et al. 1970] that uses dynamic program-

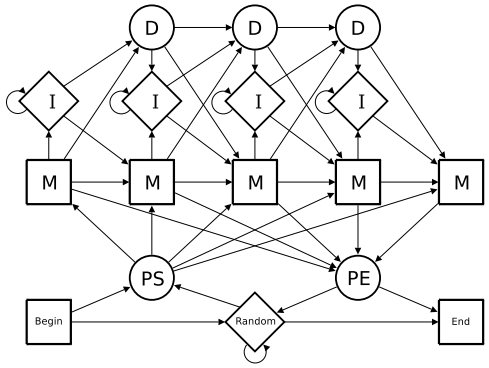

Fig. 6. An example search HMM

ming to iteratively improve the model's parameters to better represent the given training sequences. This algorithm is guaranteed to find a locally optimal set of parameters that maximizes the likelihood of the model given the training sequences. Unfortunately, parameters chosen via this method are not guaranteed to be globally optimal, and often the difference between local optima and the global optimum is substantial. Therefore, we apply simulated annealing [Kirkpatrick et al. 1983] in the Baum-Welch algorithm to decrease the risk of not progressing out of a local optimum. After this algorithm has converged, we apply Viterbi training [Vogel et al. 1996] to the resulting model to further refine its parameters for use in searching streams of packets for the given target phrase. While this last step is not guaranteed to find an optimal set of parameters, it does maximize the contribution of the most likely sequences of states to the model's likelihood, and it is widely used in bioinformatics applications for training the models used in searching protein databases [Durbin et al. 1999].

3.1.2 *Searching with a Profile HMM.* In an encrypted VoIP call, packets for the target phrase will be surrounded by packets that comprise the rest of the conversation. To isolate the target phrase from its surroundings, we add five new states to the standard profile HMM to create a search HMM, which is shown in Figure 6. The most important new state is the *Random* state, shown as a diamond in the figure because, like the Insert states, it emits packets according to a uniform or other "random" distribution. When we search a stream of packets, the Random state will match packets that are not part of the phrase of interest, and the states in the profile part of the model will match the packets in the target phrase. Two new silent states, called the *Profile Start* and *Profile End* states, are shown in Figure 6 as circles. They allow for transitions between the Random state and the profile part of the model. Because we want to find only instances of the entire target phrase, transitions from the Profile Start state are weighted such that the transition to the Match state in the first position is much more likely than the others.

To find instances of our target phrase in the sequence of packets from a VoIP conversation, we use the Viterbi algorithm [Viterbi 1967] to find the most likely sequence of states in the model to explain the observed packet sizes. Each subsequence of states which belong to the profile part of the model is called a *hit*, and is

potentially an instance of the target phrase. To evaluate the goodness of each hit, we compare the likelihood of the packet lengths given the profile model, versus their likelihood under the overall distribution from the Random state. More formally, we calculate the log odds score for a hit consisting of packet lengths $\ell_i, ..., \ell_j$, as

$$score_{i,j} = \log \frac{P(\ell_i, ..., \ell_j | Profile)}{P(\ell_i, ..., \ell_j | Random)} \quad (1)$$

Intuitively, this score tells us how well the packets match our model, and we discard any hit whose score falls below a given threshold. We return to how to set these thresholds in Section 4.7.

3.2 Synthesizing Training Data

Thus far, we have made the simplifying assumption that the adversary could build her models using several audio recordings of each word or phrase she wanted to detect. However, in practice, this assumption is far from realistic. Because of the distribution of words in natural language, even in very large corpora, there will be many words that occur only a few times, or not at all. The speech recognition community has developed efficient techniques for constructing word models without the need for labeled training examples of every word. In this section, we show how similar strategies can be applied to our task of spotting words in encrypted VoIP, even when the eavesdropper has never actually heard any of the words in the target phrase.

The techniques in this section rest on the idea that all spoken words in a language are formed by concatenating phonemes, much like words in written language are formed by making strings of letters. In a phonetic acoustic model of speech (c.f., Chapter 3 of Jelinek's text on speech recognition [Jelinek 1998]), small, profile-like HMMs are trained to represent the sounds that correspond to each phoneme. Then, to construct a word HMM, the HMMs for the phonemes used to pronounce the word are concatenated to form a long chain of states that represents the sequence of sounds in the word. Similarly, phrase HMMs are constructed by concatenating word models. Typically, the sequence of phonemes used to pronounce each word is taken from a phonetic pronunciation dictionary, such as PRONLEX [Kingsbury et al. 1997], although they may also be taken from the pronunciatons given in a standard English dictionary. Because these pronunciation dictionaries are relatively easy to create and can be stored as plain text files, it is much easier and cheaper to obtain a large-vocabulary pronunciation dictionary than to obtain a corpus of speech recordings for the same words.

3.2.1 *Building word models from phonemes.* Rather than concatenating independently trained HMMs, which are not robust to variations in pronunciation and dialect, we instead choose to use a heuristic that simultaneously retains the simplicity and efficiency of the basic profile HMM topology and the techniques outlined in the previous section, yet captures a wide range of pronunciations for each word. We use a phonetic pronunciation dictionary, together with a library of examples of the packet sequences that correspond to each phoneme, to generate a *synthetic* training set for the phrase in question. Then, using this synthetic training set in place of actual instances of the phrase, we can train a profile HMM and use it to

search VoIP conversations just as described in Section 3.1. This novel approach affords us great flexibility in finding an essentially unlimited number of phrases.

To generate one synthetic sequence of packets for a given phrase, we begin by splitting the phrase into a list of one or more words. For each word in the list, we replace it with the list of phonemes taken from a randomly-selected pronunciation of the word from our phonetic pronunciation dictionary. For example, given the phrase "the bike", we look up "the" and "bike" in our pronunciation dictionary and get the phonemes "*dh ah*" and "*b ay k*", giving us a sequence of 5 phonemes: "*dh, ah, b, ay, k*". Then, for each of the phonemes in the resulting list, we replace it with one example sequence of packets sizes taken from our library for the given phoneme.

3.2.2 *Improved Phonetic Models.* Because the sounds produced in a phoneme can vary significantly depending on the phonemes that come immediately before and immediately after, it is essential that we estimate packet distributions based on the diphones (pairs of consecutive phonemes) or triphones (three consecutive phonemes), rather than the individual phonemes in the phrase. To do so, we start by grouping the phonemes in the phrase into groups of three, so that the triphones overlap by one phoneme on each end. For example, from our sequence of phonemes

$$dh, ah, b, ay, k$$

we get the triphones

$$(dh, ah, b), (b, ay, k)$$

We then check the resulting list of triphones to make sure that we have a sufficient number of examples in our library for each triphone in the list. If the library contains too few examples of one of the triphones, we split it into two overlapping diphones. Returning to our example, if we have no samples of the triphone (dh, ah, b), we replace it with the diphones (dh, ah) and (ah, b), giving us the sequence

$$(dh, ah), (ah, b), (b, ay, k)$$

Similarly, we replace any diphones lacking sufficient training data with single phonemes. As this small example illustrates, this technique allows us to use a better phonetic model for sequences of phonemes for which we have several examples in our library, yet allows a great deal of flexibility for combinations of words or sounds that we have not seen before. If, for instance, the training corpus in our example does not contain "the bike", but it does have examples of people saying "the" (**dh, ah**), "a bird" (**ah, b***, er, d*), and "bicameral" (**b, ay, k***, ae, m, ax, r, ax, l*), we can still derive a good model for the packets that will occur when a VoIP caller says "the bike".

Thus, to identify a phrase without using any examples of the phrase or any of its constituent words, we apply this concatenative synthesis technique to generate a few hundred synthetic training sequences for the phrase. We use these sequences to train a profile HMM for the phrase and then search for the phrase in streams of packets, just as in the previous section. An overview of the entire training and detection process is given in Figure 7.

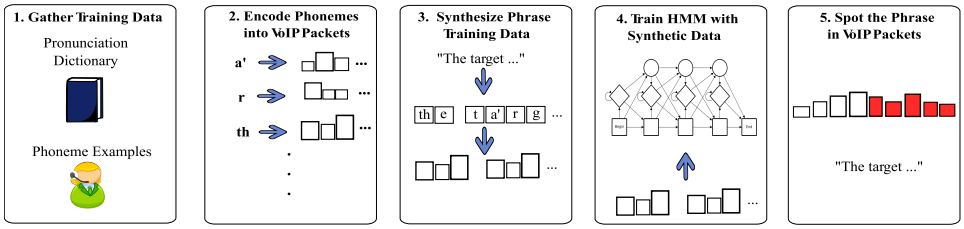

Fig. 7. Overview of the training and detection process

4. EMPIRICAL EVALUATION

To evaluate our phrase spotting technique, we focus our efforts on assessing the impact of various features of the underlying audio on phrase spotting performance, and examine the ability of an attacker to detect the presence of phrases within an encrypted packet stream. In these experiments, we use audio recordings from the TIMIT continuous speech corpus [Garofolo et al. 1993], one of the most widely used corpora in the speech recognition community. The TIMIT corpus contains 6,300 phonetically rich English sentences spoken by a total of 630 people—462 speakers randomly selected by the corpus' creators as a training set and the remaining 168 speakers designated as a test set. Speakers in the data set include males and females with eight distinct regional dialects from across the continental United States. Both the test and training sets include all gender and region combinations.

One of the most appealing features of TIMIT for our evaluation is that it includes time-aligned phonetic transcriptions of each sentence that denote the start and end of each phoneme. After encoding the audio in the training set with Speex in wideband VBR mode, we use these phonetic transcriptions to build our library of packet sequences that correspond to each phoneme, diphone, and triphone in the training set. Although TIMIT includes a primitive pronunciation dictionary containing pronunciations for each word in the corpus, the included pronunciations were originally taken from an old version of Merriam-Webster's Pocket Dictionary, and thus may represent "proper" American English rather than colloquial speech. Therefore, we also use the phonetic transcriptions for the training sentences to build up an empirically-derived pronunciation dictionary based on the way the speakers say each word in the training data. For increased coverage in our empirical dictionary, we also include pronunciations from the PRONLEX dictionary, which were derived in a similar to our empirically-derived dictionary by using the CALLHOME telephone speech corpus [Kingsbury et al. 1997].

4.1 Experimental Setup

To evaluate the effectiveness of our phrase spotting techniques, we use the TIMIT training data to build HMMs to search for 122 target sentences. Specifically, we sample uniformly over the available pronunciations for each of the words in the sentence to determine the sequence of phonemes to synthesize. We then use the synthesis technique described in Section 3.2 to generate the sequence of packets associated with those phonemes. Since there are many possible pronunciations and packet sequences associated with each of the words in the sentence, we repeat

this process 400 times for each sentence to create a diverse collection of training samples. This data is then used to train the profile HMMs created using the HMMER software package [Eddy 2009]. In our experiments, we use the default initial transition and emission probabilities for HMMER, except in the case of the Random state where we use the unigram distribution of packet sizes found in our training data.

In order to produce realistic testing data, we simulate VoIP conversations for each of the speakers in the TIMIT test set by taking two copies of each of the given speaker's sentences, and concatenating all of them in a random order. By randomly placing multiple copies of the same phrase within a simulated conversation, we can ensure that our ability to spot the phrase is not dependent on the context of the preceding or succeeding phrases in the conversation. We create five of these simulated conversations for each speaker to minimize any impact of the sentences' location in the conversation on the performance of our algorithms.

We then encode the simulated conversations with wideband Speex in VBR mode and use the profile HMMs to search for instances of each phrase in the resulting stream of packet lengths. From the Viterbi alignment of the packet lengths to the phrase HMM, we get the subsequence(s) of packets indicating potential hits for the phrase, with log odds scores for each. Subsequences with scores above a given threshold are considered definitive hits, and each hit is labeled as a true positive only if it contains *all* of the words for the given phrase. Any definitive hit which does not contain all words in the phrase is considered a false positive.

We adapt standard metrics from the information retrieval community to assess the effectiveness of our approach. Let TP_t, FP_t, and FN_t be the number of true positives, false positives, and false negatives achieved when operating with threshold t. Then, the *precision* at t is defined as $\mathsf{prec}_t = TP_t/(TP_t + FP_t)$ and measures the probability that a reported match is correct. We also use *recall*, defined as $\mathsf{recall}_t = TP_t/(TP_t + FN_t)$, as the probability that the algorithm will find the phrase if the phrase is indeed contained within the ciphertext. Ideally a search algorithm would exhibit precision and recall both close to 1.0.

To assess the accuracy of our approaches under different parameters, we compute recall and precision over a variety of thresholds. An intuitive way to derive the threshold for a given model would be to use the average log odds score (Equation 1) of the training sequences. However, since the log odds score is proportional to the length of the phrase, we cannot directly compare the performance of models for different phrases at the same log odds score. Therefore, to compare accuracy between models for different phrases, we set the threshold for each model to be some fraction of the model's log odds score on the training data. Explicitly, for each phrase p, let σ_p be the average log odds score observed during training for the model m_p. σ_p will be proportional to the length of m_p. For a multiplier $\delta \in [0, 2]$ we set the testing threshold $t_p = \delta \times \sigma_p$, and compute the average precision and recall at multiplier δ using TP_{t_p}, FP_{t_p}, and FN_{t_p} for each phrase p in our testing set. We can then examine how precision relates to recall by plotting average precision versus average recall at each value of δ.

With these comparison metrics at hand, we can now proceed to analyze the accuracy of our approach. First, we take an analytical approach and examine the

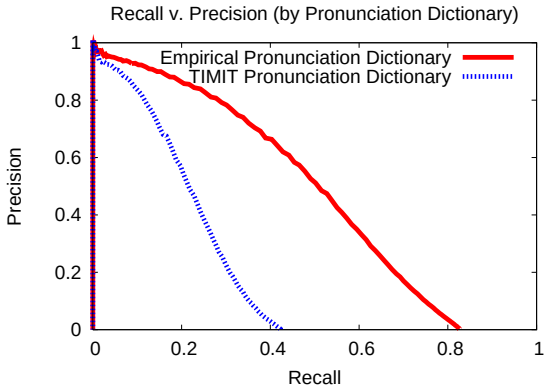

Fig. 8. The impact of pronunciation dictionary choice.

performance of our method over a range of thresholds to study the impact of the pronunciation dictionary, the speaker's gender, background noise, and sampling rate. We further examine our results under less stringent definitions of true and false positives to better understand the realistic impact of these results; specifically, we look at the performance with respect to individual packets. Finally, we assume the viewpoint of an attacker and empirically estimate a specific threshold for each phrase.

4.2 The Importance of Accurate Pronunciation Dictionaries

In order to build a model for a phrase, we first must know the phonemes that comprise the phrase. While the TIMIT pronunciation dictionary describes which phonemes *should* appear in each phrase, we believe that these phonemes may not necessarily be representative of colloquial speech. To verify this hypothesis, we compare the accuracy of our approach using models built with this pronunciation dictionary and with models built with our empirically derived dictionary. The results of this analysis are presented in Figure 8.

Clearly, the quality of the pronunciation dictionary is critical to the success of our phrase spotting technique. With the default TIMIT pronunciations, we achieve equal recall and precision at around 0.28. However, using the more realistic pronunciation dictionary, we simultaneously achieve recall of 0.50 and precision of 0.51. In other words, on average we are able to find 50% of the instances of the phrases of interest, and when the algorithm indicates a match, there is a 51% chance that the flagged packets do indeed encode the given phrase. These results are especially disconcerting given that the conversation was *encrypted* in order to prevent an eavesdropper from recovering this very information. In light of these results, we perform the remaining experiments using only the empirically derived pronunciation dictionary.

4.3 Gender Differences

To further explore the impact of training data composition on the accuracy of our approach, we divided the testing and training set by gender and performed the

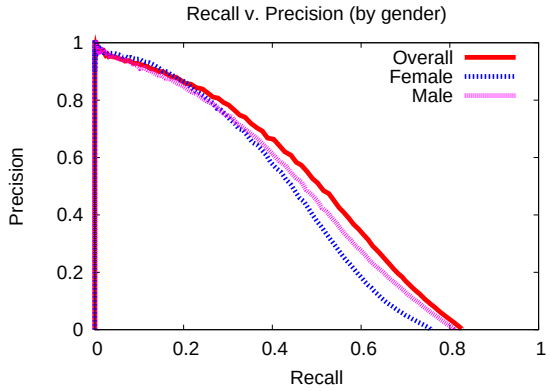

Fig. 9. Recall and precision by gender.

same test. The hope here is that we might be able to eliminate any sort of gender variation to improve models. In Figure 9, we see that this is not the case. It appears that any improvement due to gender-specific training is more than offset by the decrease in the size of the training set. Our accuracy for female speakers, who comprise only 30% of the data set, decreases more than for the male speakers, for whom we can still spot phrases with nearly unchanged accuracy.

4.4 Robustness to Noise

We also evaluate the impact of noise on our ability to identify phrases. For this test, we add *pink noise* to the simulated conversations in the TIMIT test data. We chose pink noise, rather than white noise or any number of background sounds (metal pots and pans clanging, a baby crying, etc.), because the energy is logarithmically distributed across the range of human hearing. This makes pink noise much more difficult for the codec's noise removal algorithm to filter, and therefore should influence the choice of bit rates in the packets. Furthermore, the use of such additive noise generation techniques is common practice for exploring the impact of noise on speech recognition methods (e.g., [Tibrewala and Hermansky 1997; Okawa et al. 1998; Junqua et al. 1994]).

We experimented with three additive noise scenarios: 90% sound to 10% noise, 75% to 25%, and 50% to 50%. With 10% noise, the recordings sound as if they were transmitted over a cell phone with poor reception, and with 50% noise it is almost impossible to tell what is being said. Figure 10 shows the results for these experiments. Notice that with 10% noise, we are still able to achieve recall of .39 and precision of .40. Even with 25% noise, we can still achieve recall and precision of .22 and .23, respectively. These results show that as long as the quality of the voice channel is reasonable, the attacker can still identify an alarming number of phrases.

4.5 Wideband versus Narrowband

The Speex codec that we use for our evalution, in addition to offering the option of variable bit-rate compression, also offers different modes of operation in order

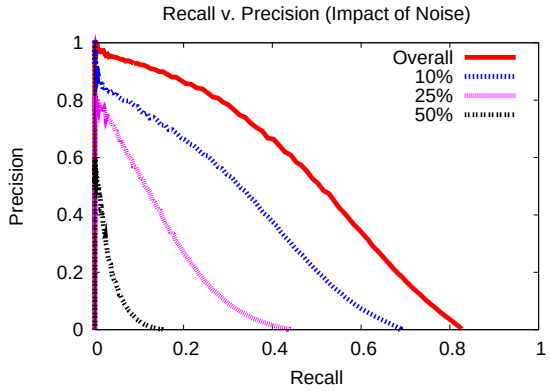

Fig. 10. Precision and recall when presented with noisy data.

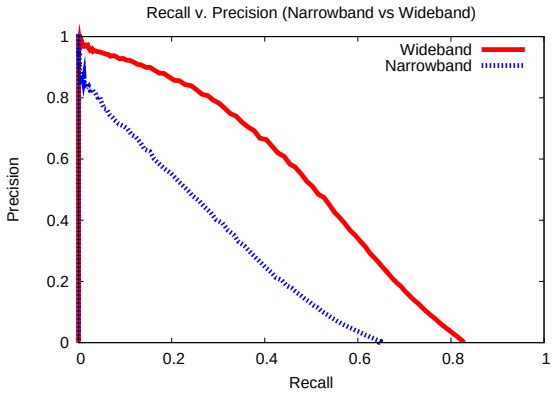

Fig. 11. Precision and recall for narrowband (8kHz) data.

to handle audio inputs of varying fidelity. Its narrowband mode operates on an input signal sampled at 8kHz, and aims to provide telephone-quality speech with low bandwidth requirements. Wideband mode operates on input signals sampled at 16kHz, with the goal of improved audio quality at the cost of increased traffic volume.

Fortunately, the TIMIT corpus comes to us sampled at 16kHz. To compare our search HMM's performance on narrowband-encoded speech to our existing results on wideband data, we downsample the TIMIT recordings to 8kHz and compress them with Speex in narrowband mode. Using the resulting narrowband packets, we synthesize training data, build phrase models, and run the HMM search just as we did for the wideband data. In Figure 11, we see that our search algorithm is significantly less effective on narrowband data. The recall and precision results are in fact roughly similar to those we achieve on wideband data with 10% noise. This change in performance is due primarily to the decrease in the number of bit rates from twenty one in wideband mode to only nine in narrowband mode. In effect,

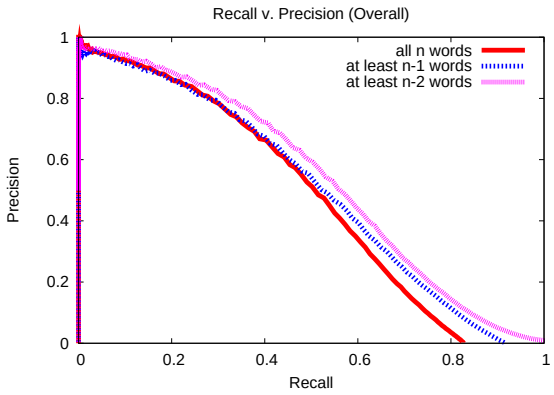

Fig. 12. Precision and recall when relaxing the requirement that all words be found in a hit.

this reduction in bit rates requires that more phonemes be grouped into the same bit rate, thereby reducing the information about the underlying audio represented in the packet sizes.

4.6 Relaxing the Criteria for Exact Matches

Due to our strict definition of a true positive, the aforementioned analysis might understate the practical performance of our technique. For instance, in several of the false negatives, the reason why the target sentence was not identified was because the model failed to match one or two of the words at the beginning or end of the phrase. In general, this behavior is due to limited training data to represent the appropriate phonetic context at the beginning and end of phrases.

Therefore, to better gauge the potential improvements we can attain if our requirement for a true positive were slightly relaxed, we report the recall and precision achieved when the search HMM is allowed to miss a small fraction of the words in a phrase, which is illustrated in Figure 12. Both recall and precision increase in this more relaxed setting, although the improvement is greater for recall than precision. When we consider reported matches that contain at least $n - 2$ of the n words in the phrase as true positives, our approach achieves recall of .55 and precision of .53. Compared to our original, stricter classification, this represents an increase of 9% to recall and 4% to precision. These results show that our definition of a true positive is an important part of the explanation for why the technique fails on some phrases. Note that when we limit ourselves to all but two of the words in a reported match to be a true positive, we can attain 100% recall for some phrases.

Further, we can quantify the performance of our method in terms of the proportion of *packets* that are correctly identified as being part of a given phrase. From the annotations on the TIMIT data, we can determine whether each packet contains audio for our target phrase. We then compare the sequences of packets flagged by our HMM search algorithm with this ground truth to determine the true positives and false positives for the algorithm with respect to packets. The precision versus recall curve based on per-packet results, shown in Figure 13, indicates that our performance improves slightly over that of our per-word results. This alternate

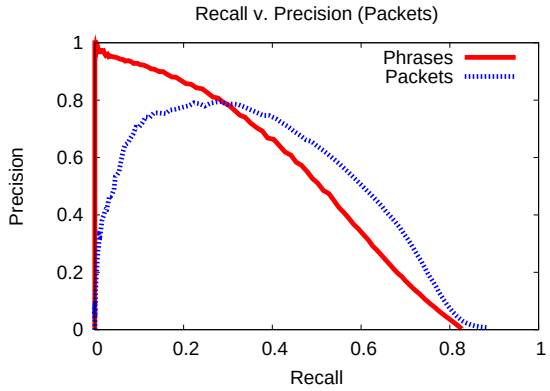

Fig. 13. Precision and recall when measuring hits on a per-packet basis

view of our results achieves a precision and recall of 0.6. As a whole, these results indicate that an attacker can very accurately identify phrases if they are willing to tolerate loose matches.

4.7 An Attacker's Point of View

Until now, we studied the success of our techniques across a wide range of thresholds. An attacker, on the other hand, would need to pick a *single* threshold in advance. Unfortunately for the attacker, picking an optimal threshold in such cases is an open research problem. Therefore, to explore the problem of threshold selection, we discuss a technique to estimate a good threshold, and the resulting expected performance.

As mentioned in Section 3, for a phrase p, the average log odds score σ_p that is observed during the training of model m_p is roughly indicative of how well the model will be able to perform in practice. Loosely speaking, if σ_p is large, then the model will exhibit high true positive rates. We use this observation to our advantage when selecting the attack threshold t_p. That is, we empirically estimate t_p as a linear function of σ_p, setting $t_p = \delta \times \sigma_p$, where δ is a multiplier that maximizes the "quality" of the search algorithm. To complete our task of selecting a threshold we must then solve two problems: (*1*) select a general function that defines the "quality" of the search algorithm at a specific threshold; and (*2*) choose a way to estimate the δ that maximizes quality.

While we could define the "quality" at threshold t_p as either recall_{t_p} or $\mathsf{precision}_{t_p}$, neither metric is appropriate for this task. Instead, to achieve a good balance of precision and recall, we define the quality of a search algorithm at threshold t to be the difference between the number of true positives and the number of false positives at t_p: $TP_{t_p} - FP_{t_p}$.

If the adversary has access to a relatively small number of recorded phrases, she can build search HMMs for them and use the performance of these models to derive a good value of δ for use in setting the thresholds for other phrases that she really wants to search for. We use *leave-out-k* cross validation to estimate her chances of success using the TIMIT testing data. In each of several iterations, we

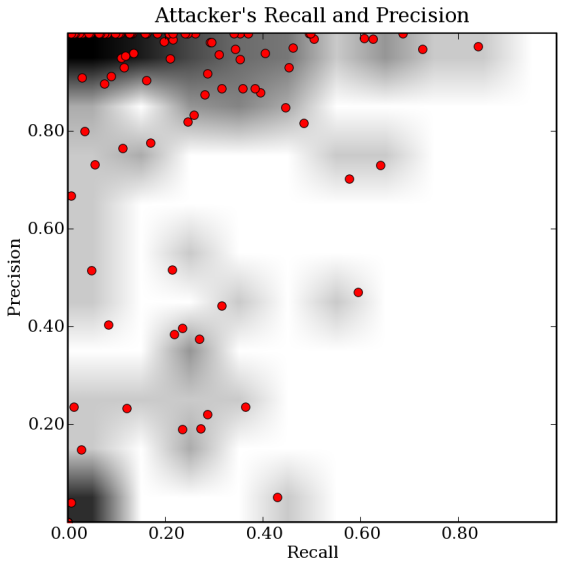

Fig. 14. Attacker's recall and precision for 122 phrases.

select k phrases $(\tilde{p}_1, \ldots, \tilde{p}_k)$ at random from the testing set and find the thresholds $t_{\tilde{p}_1}, \ldots, t_{\tilde{p}_k}$ that maximize the difference in true positives and false positives for each phrase. We set $\delta_{\tilde{p}_i} = t_{\tilde{p}_i}/\sigma_{\tilde{p}_i}$ for each $i \in [1, k]$, and set δ to be the average over $\delta_{\tilde{p}_i}$. Then, for each phrase p in the remainder of the test set, we estimate our maximizing threshold for p to be $t_p = \delta \times \sigma_p$, and calculate the recall and precision when searching for phrase p at threshold t_p.

Setting k to be 1/4 of our testing set, this technique achieves mean recall and precision rates of .32 and .75, respectively. Given that our original averages were .50 and .51, it seems that our estimation technique is somewhat conservative, selecting thresholds that are higher than optimal. The values of recall and precision achieved for each phrase using our threshold selection algorithm are presented in Figure 14. Each of the points indicates the recall and precision for one of the 122 phrases in our test set. Because simple scatter plots often plot many points on top of one another, we also vary the background color to indicate the density of the points in each area of the graph. Dark backgrounds indicate high density, and light backgrounds indicate areas of low density. While this algorithm is not optimal, its recall is often above 40%, and we can recognize most of the phrases with precision greater than 80%. We believe this shows concretely that an attacker with access to only population statistics, and the ciphertext of a VBR encoded and encrypted VoIP conversation has almost a one in three chance of finding a phrase of her choice!

5. ANALYSIS OF RESULTS

While our phrase spotting approach performs well on average, there are several phrases that we excel at uncovering and many that we are simply unable to find.

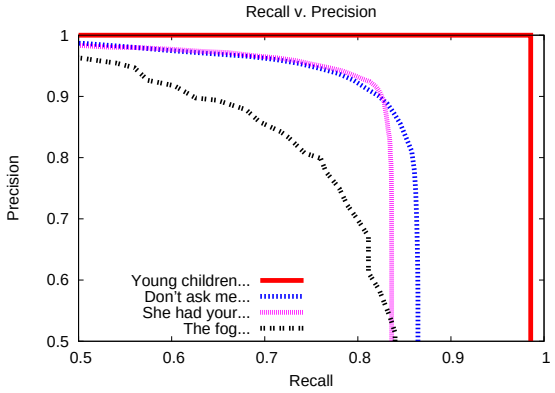

Fig. 15. Precision and recall for several interesting phrases.

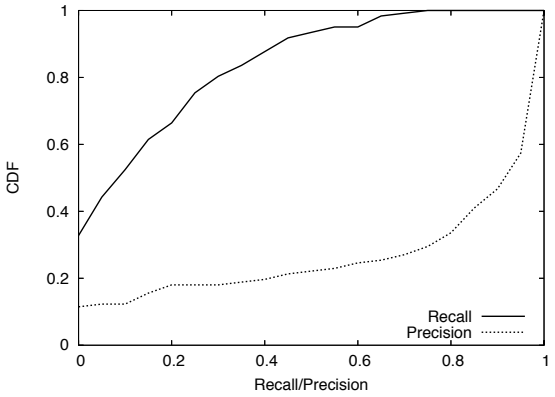

Fig. 16. Cumulative distribution function for phrase recall and precision.

Figure 15 shows the precision and recall for four phrases that our approach does particularly well in finding, and Appendix B provides an expanded listing of the best and worst phrases in our evaluation. The phrase that exhibited the best accuracy was *"Young children should avoid exposure to contagious diseases"*, on which we achieve precision of 1.0 and recall of .99. Several other phrases were discovered with precision ranging from .92 to .84 and recall ranging from .82 to .72. There were also a number of phrases that we are unable to detect with sufficient accuracy. Of the 122 phrases tested, ten phrases yield an average precision and recall of zero. An example of one of these phrases is *"Straw hats are out of fashion this year."* As shown in the cumulative distribution function in Figure 16, approximately half of the phrases we test have recall of less than 0.1, and yet most phrases yield a precision exceeding 0.8. These results naturally raise questions about which features of the phrases make them easily detectable, and why certain phrase are so difficult to detect.

To investigate this disparity, we examine a number of features of the phrases.

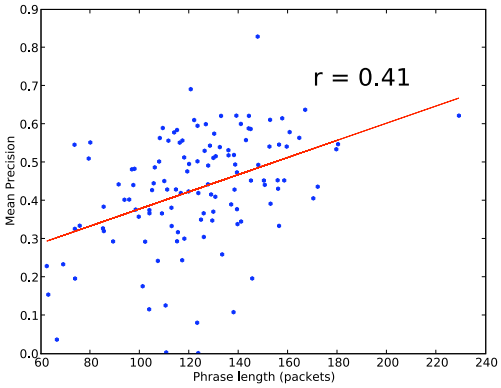

Fig. 17. Phrase length versus mean precision for our 122 test phrases.

These features include the length of the phrase, its phonetic composition, the speakers of the phrase, and its distribution of packets sizes and phonemes. Of these features, only the length of the phrase, and the overall performance of the speakers of the phrase correlated with either recall or precision. Specifically, we have found that phrase length correlates with the phrase's precision, and that the average true positive rate of the speakers of the phrase correlates with both its recall and precision. In this section, we discuss these two features in greater detail to glean an understanding of when our proposed attack might be most dangerous to the privacy of VoIP calls.

For each feature that we examine, we calculate the sample correlation coefficient (i.e., Pearson's r) between the feature, and the recall (respectively, precision). Intuitively, r captures the strength of the linear dependence between two variables, and r^2 indicates the amount of variance in the values of the variables that can be attributed to that dependence. In addition, we test the significance of the correlation by using the Student's t-test, which is defined as:

$$t = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}}$$

where r is the correlation coefficient, and $n = 122$ is the number of phrases in our test. The t-test indicates the level of confidence that we can place in the fact the dependence between the feature and recall (or, precision) is linear. With our test of 122 phrases, we require a value of $t > 2.36$ for 99% confidence in our correlation measure.

5.1 Phrase Length

The first feature that we discuss is the length of the phrase, which we define as the average number of packets that make up the phrase when it is spoken in a VoIP call. Figure 17 show the average phrase length versus the average precision for the phrase calculated over all thresholds. Here, we find a moderate correlation of $r = 0.41$ between the phrase's length and its precision. This indicates that $r^2 = 16.8\%$ of

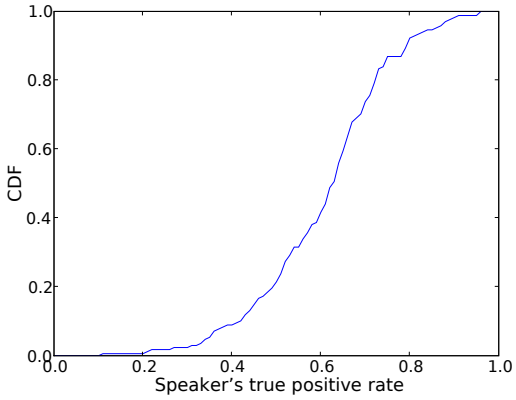

Fig. 18. Cumulative distribution function of per-speaker true positive rates

the variation in the precision is caused by the length of the phrase. When we apply the Student's t-test, we find that $t = 4.92$, which indicates that there is a linear relation between the phrase length and precision with 99% confidence. There is, however, no correlation between phrase length and recall ($r = 0.04$).

The relationship between phrase length and precision is intuitive when we consider the possible effects of a long phrase on our search HMM. Longer phrases are easier to spot reliably because, with each additional packet, the search HMM collects additional evidence that it has really found an instance of the target phrase. Likewise, short phrases are difficult to spot reliably because it is much easier for short patterns of packets to occur by chance as a result of other speech that is not an instance of the target phrase. Therefore, as the length of the phrase increases, the number of false positives from the search HMM decreases, causing the detector's precision to increase.

5.2 Problem Speakers

In addition to phrase length, we also found that the performance of our technique depends at least in part on the individual speaker. That is, there are some speakers in the dataset for whom our phrase spotting techniques work exceptionally well regardless of the phrase, and some for whom we are not able to spot phrases accurately at all. To demonstrate this, we calculated the average true positive rate that our algorithm achieves for each speaker in the test set as the fraction of the speaker's utterances that our algorithm correctly detects. Figure 18 shows the cumulative distribution function of our per-speaker true positive rates across all speakers in the dataset.

Then, we quantify the dependence between our performance on individual speakers and our ability to recognize the phrases in the dataset. For each phrase, we calculated the average true positive rates for each user who spoke the phrase to derive the average speaker true positive rate for the phrase. To better identify the relationship between the speakers of a phrase and the accuracy with which we detect it, we exclude from this part of the analysis two of the 122 phrases because

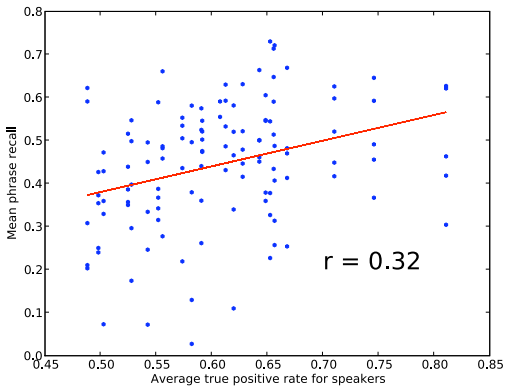

Fig. 19. Average per-speaker true positive rate versus recall

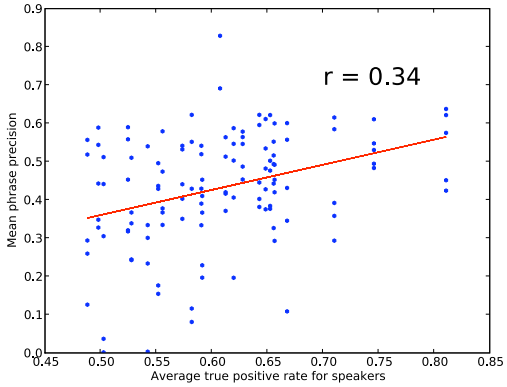

Fig. 20. Average per-speaker true positive rate versus precision

they were spoken by everyone in the data set. The remaining 120 phrases were each spoken by 7 people.

Figure 19 shows the average speaker true positive rate for the phrase versus our average recall for the phrase across all threshold levels. The sample correlation coefficient for this correlation is $r = 0.32$, which indicates that 10.2% of the variation in recall is due to the effects of the speakers. The t-test for these variables shows that a linear dependence exists between the two with 99% confidence ($t = 3.67$). Furthermore, there is also a correlation between the speaker's average true positive rate and precision ($r = 0.34$), and again the t-test suggests a linear dependence with 99% confidence ($t = 3.93$). The correlation between precision and the average speaker true positive rate is shown in Figure 20.

The diference in performance among users can be attributed to the fact that our synthetic generation of training data trains the search HMM to find phrases as they would be spoken by the average user in our TIMIT corpus. Thus, many

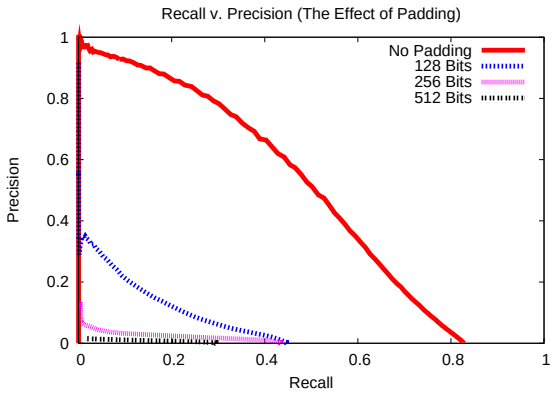

Fig. 21. Impact of padding on the search algorithm.

users with unusual dialects or acoustic features may not be well represented within our training data, leading to misclassification of their speech. Unfortunately, the relatively small size of the TIMIT corpus makes it difficult to further examine the acoustic features of the users to discern those features which impact our techniques the most. Additionally, it is reasonable to assume that there are several higher order interactions among features of the phrases and the speaker's acoustics that more directly affect performance, but which are not easily observable in the data.

6. TECHNIQUES FOR MITIGATION

Given our phrase spotting methodology's reliance on packet sizes and their ordering, there are two apparent ways of defeating our technique: (*1*) altering the order of packets, or (*2*) quantizing packet sizes to reduce the information about the underlying audio. Due to the real-time nature of VoIP, re-ordering packets may reduce the perceived audio quality of the call. Unfortunately, this reduction in quality may be difficult to measure after the fact and even harder to predict *a priori*. For example, the current ITU standard for objective voice quality measurement [Rix et al. 2001] has not been validated for use with noise reduction or echo cancellation, both of which are frequently used with codecs like Speex. Therefore, in the following section we limit our examination of mitigation strategies to packet padding.

To explore the tradeoff between padding and search accuracy, we encrypted both our training and testing data sets to multiples of 128, 256 or 512 bits and applied our approach. The results are presented in Figure 21. The use of padding, in this case, is quite encouraging as a mitigation technique. When padding to multiples of 128 bits, we achieve only 0.15 recall at 0.16 precision. Increasing padding so that packet sizes are multiples of 256 bits gives recall of .04 at .04 precision. Finally, by using a block size of 512 bits, we ensure that all packet sizes are exactly the same, and consequently our recall and precision are reduced to zero.

Nevertheless, padding to 128, 256, and 512 bit blocks results in overhead of 8.81%, 16.5%, and 30.82%, respectively. We also note that these bandwidth estimates are likely lower than the overhead incurred in practice, because as Chu notes [Chu 2003], in a two-way call each participant is idle roughly 63% of the time, enabling

the transmission of many smaller packets. Our testing data, on the other hand, is comprised of continuous speech, and so the smaller packets that indicate silence are less prevalent.

7. RELATED WORK

In 1982, Simmons and Holdridge [Simmons and Holdridge 1982] highlighted the shortcomings of an early design for encrypting voice traffic using a semantically-insecure version of RSA. They showed that an adversary with knowledge of the recipient's public key could recover the audio from an encrypted conversation by pre-computing ciphertexts for a moderate number of sounds and then observing when the same ciphertexts were transmitted.

More recently, the increasing popularity of Internet telephony has encouraged several studies of VoIP and security. Wang et al. [Wang et al. 2005] proposed a method of tracking VoIP calls across anonymizing networks, like ToR [Dingledine et al. 2004], through the use of packet timing as a watermark. Verscheure et al. [Verscheure et al. 2006] then presented an entirely passive method for identifying the endpoints of an anonymized VoIP call by observing patterns in the packet stream due to the encoder's voice activity detection. Work by Pelaez-Moreno et al. [Pelaez-Moreno et al. 2001] and Aggarwal et al. [Aggarwal et al. 2005] has examined the problem of speech recognition from compressed VoIP. Finally, we have shown in earlier work that it is possible to identify the language spoken by the callers in a VoIP conversation using only the sizes of the encrypted packets [Wright et al. 2007].

Additionally, there is a growing body of work focusing on inference of sensitive information from encrypted network connections using packet sizes and timing information. Sun et al. [Sun et al. 2002] and Liberatore and Levine [Liberatore and Levine 2006] have shown that it is possible to identify web pages traversing encrypted HTTP connections (e.g., SSL) using only the number and size of the encrypted HTTP packets. In a similar vein, Saponas et al. [Saponas et al. 2007] proposed a method to identify videos played over an encrypted network channel using the total size of the packets transmitted in a short window of time. Additionally, packet inter-arrival times have been used to infer keystrokes within encrypted SSH sessions [Song et al. 2001].

The techniques presented in this paper are heavily influenced by the speech recognition community and its established methods for wordspotting. The most widely accepted method of wordspotting in continuous speech data takes advantage of hidden Markov models (HMMs) trained on acoustic features of complete words (e.g., [Rohlicek et al. 1989; Wilpon et al. 1990]), or the composition of phonemes into words (e.g., [Rohlicek et al. 1993; Rose and Paul 1990]). For HMMs trained on whole-word acoustic data, detection rates can reach upwards of 95%, but such approaches are inherently limited to relatively small vocabularies where there is an abundance of training data available for each word. On the other hand, phonetically-trained acoustic HMMs are able to spot any word based solely on its phonetic transcription and acoustic data for the phonemes. However, detection rates for these phoneme-based systems tend to fall to between 75% and 85% due to the difficulty of capturing word-specific pronunciation variability. At a high level, our VoIP phrase spotting technique uses phonetically-trained HMMs, but the

specifics of their use are drastically different from that of typical speech since we do not have access to the underlying acoustic data. Despite the coarse nature of the information gained from encrypted VoIP packet sizes, our performance is not significantly worse than that of early wordspotting methods in speech.

8. CONCLUSION

Voice over IP technologies are rapidly replacing traditional telephony despite the fact that relatively little is known about the security and privacy implications of its adoption. Although previous work has shown that the combination of VBR compression with length-preserving encryption leaks information about VoIP conversations [Wright et al. 2007], the true extent of this information leakage is not fully understood. In our earlier work on phrase spotting [Wright et al. 2008], we showed that this information leakage is far worse than originally thought by using a profile hidden Markov model trained using speaker- and phrase-independent data. Our approach detected the presence of some phrases within encrypted VoIP calls with recall and precision exceeding 90%. On average, our method achieved recall of 50% and precision of 51% for a wide variety phonetically rich phrases spoken by a diverse collection of speakers.

In this paper, we expanded upon these initial results to better understand the impact of noise, dictionary size, gender, and audio quality on the performance of our techniques. Moreover, we also examine the performance of our phrase spotting methodology with partial matching, which shows an increase in both the recall and precision. Finally, we examined the underlying reasons behind the performance of our techniques, and found that the primary indicators of performance for our approach are the length of the phrase and the speaker of the phrase.

Overall, the results of our study show that an attacker can spot a variety of phrases in a number of realistic settings, and underscores the danger in using the default encryption transforms of the SRTP protocol – none of which specify the use of padding [Baugher et al. 2004]. Given that many existing VoIP implementations, including Skype [Zimmerman 2008], make use of variable bit rate encoding methods, these attacks reflect a potentially serious violation of user privacy. Although padding could introduce inefficiencies into real-time protocols, our analysis indicates that it offers significant confidentiality benefits for VoIP calls. An important direction of future work in this area focuses on the development of padding techniques that provide an appropriate balance between efficiency and privacy.

REFERENCES

Aggarwal, C., Olshefski, D., Saha, D., Shae, Z. Y., and Yu, P. 2005. Csr: Speaker recognition from compressed VoIP packet stream. In *Proceedings of the IEEE International Conference on Multimedia and Expo, 2005.* 970–973.

Baugher, M., McGrew, D., Naslund, M., Carrara, E., and Norrman, K. 2004. The secure real-time transport protocol (SRTP). RFC 3711.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics 41,* 1 (February), 164–171.

Chu, W. C. 2003. *Speech Coding Algorithms.* John Wiley and Sons.

Dempster, A. P., Laird, N. M., and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society 39,* 1, 1–38.

Dimolitsas, S., Sherif, M., South, C., and Rosenberger, J. R. 1993. The CCITT 16 kbit/s speech coding recommendation G.728. *Speech Communications 12,* 2, 97–100.

Dingledine, R., Mathewson, N., and Syverson, P. 2004. Tor: The Second-Generation Onion Router. In *Proceedings of the* 13^{th} *USENIX Security Symposium*. 303–320.

Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. 1999. *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press.

Eddy, S. 1995. Multiple alignment using hidden Markov models. In *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*. 114–120.

Eddy, S. 2009. HMMER: Biosequence Analysis Using Profile Hidden Markov Models. `http://hmmer.janelia.org`.

ETSI/GSM 1991. GSM Full Rate Speech Transcoding. GSM Recommendation 06.10.

Gardner, W., Jacobs, P., and Lee, C. 1993. QCELP: A variable bit rate speech coder for CDMA digital cellular. *Speech and Audio Coding for Wireless and Network Applications*, 85–92.

Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., Pallett, D. S., Dahlgren, N. L., and Zue, V. 1993. TIMIT acoustic-phonetic continuous speech corpus. Linguistic Data Consortium, Philadelphia.

Jelinek, F. 1998. *Statistical Methods for Speech Recognition*. MIT Press.

Junqua, J. C., Mak, B., and Reaves, B. 1994. A robust algorithm for word boundary detection in the presence of noise. *IEEE Transactions on Speech and Audio Processing 2,* 3, 406–412.

Kingsbury, P., Strassel, S., Lemore, C., and MacIntyre, R. 1997. CALLHOME american english lexicon (PRONLEX). Linguistic Data Consortium, Philadelphia.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. 1983. Optimization by Simulated Annealing. *Science 220,* 4598, 671–680.

Krogh, A., Brown, M., Mian, I. S., Sjölander, K., and Haussler, D. 1994. Hidden Markov Models in computational biology: Applications to protein modeling. *Journal of Molecular Biology 235,* 5 (February), 1501–1531.

Liberatore, M. and Levine, B. 2006. Inferring the Source of Encrypted HTTP Connections. In *Proceedings of the ACM conference on Computer and Communications Security*. 255–263.

Okawa, S., Bocchieri, E., and Potamianos, A. 1998. Multi-band speech recognition in noisy environments. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 1998*.

Pelaez-Moreno, C., Gallardo-Antolin, A., and de Maria, F. D. 2001. Recognizing voice over IP: A robust front-end for speech recognition on the World Wide Web. *IEEE Transactions on Multimedia 3,* 2 (June), 209–218.

Provos, N. 2004. Voice Over Misconfigured Internet Telephones. `http://vomit.xtdnet.nl`.

Rabiner, L. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE 77,* 2 (February).

Rix, A. W., Beerends, J. G., Hollier, M. P., and Hekstra, A. P. 2001. Perceptual evaluation of speech quality (PESQ), an objective method for end-to-end speech quality assessment of narrowband telephone networks and speech codecs. ITU-T recommendation P.862.

Rohlicek, J. R., Jeanrenaud, P., Ng, K., Gish, H., Musicus, B., and Siu, M. 1993. Phonetic training and language modeling for word spotting. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 1993*.

Rohlicek, J. R., Russell, W., Roukos, S., and Gish, H. 1989. Continuous hidden Markov modeling for speaker-independent wordspotting. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 1989*. 627–630.

Rose, R. C. and Paul, D. B. 1990. A hidden Markov model based keyword recognition system. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 1990*. 129–132.

Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and Schooler, E. 2002. SIP: Session initiation protocol. RFC 3261.

Saint-Andre, P. 2004. Extensible messaging and presence protocol (XMPP): Core. RFC 3920.

Saponas, T. S., Lester, J., Hartung, C., Agarwal, S., and Kohno, T. 2007. Devices that tell on you: Privacy trends in consumer ubiquitous computing. In *Proceedings of the* 16^{th} *Annual USENIX Security Symposium*. 55–70.

Schroeder, M. R. and Atal, B. S. 1985. Code-excited linear prediction(CELP): High-quality speech at very low bit rates. In *Proceedings of the 1985 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 10. 937–940.

Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V. 1996. RTP: A transport protocol for real-time applications. RFC 1889.

Simmons, G. J. and Holdridge, D. 1982. Forward search as a cryptanalytic tool against a public key privacy channel. In *Proceedings of the IEEE Symposium on Security and Privacy*. 117–128.

Skype 2009. Skype. `http://www.skype.com`.

Song, D., Wagner, D., and Tian, X. 2001. Timing analysis of keystrokes and SSH timing attacks. In *Proceedings of the* 10^{th} *USENIX Security Symposium*.

Sun, Q., Simon, D. R., Wang, Y.-M., Russell, W., Padmanabhan, V. N., and Qiu, L. 2002. Statistical identification of encrypted web browsing traffic. In *Proceedings of the IEEE Symposium on Security and Privacy*. 19–30.

Tibrewala, S. and Hermansky, H. 1997. Sub-band based recognition of noisy speech. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 1997*. 1255–1258.

Valin, J.-M. and Montgomery, C. 2006. Improved noise weighting in CELP coding of speech - applying the Vorbis psychoacoustic model to Speex. In *Audio Engineering Society Convention*. See also `http://www.speex.org`.

Verscheure, O., Vlachos, M., Anagnostopoulos, A., Frossard, P., Bouillet, E., and Yu, P. S. 2006. Finding who is talking to whom in voip networks via progressive stream clustering. In *Proceedings of the Sixth International Conference on Data Mining*. 667–677.

Viterbi, A. J. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory IT-13*, 260–267.

Vogel, S., Ney, H., and Tillmann, C. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the* 16^{th} *Conference on Computational Linguistics*. Vol. 2. 836–841.

Wang, X., Chen, S., and Jajodia, S. 2005. Tracking anonymous peer-to-peer VoIP calls on the Internet. In *Proceedings of the* 12^{th} *ACM conference on Computer and communications security*. 81–91.

Wilpon, J. G., Rabiner, L. R., Lee, C. H., and Goldman, E. R. 1990. Automatic recognition of keywords in unconstrained speech using hidden Markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing 38,* 11, 1870–1878.

Wright, C. V., Ballard, L., Coull, S. E., Masson, G. M., and Monrose, F. 2008. Spot Me If You Can: Uncovering Spoken Phrases in Encrypted VoIP Conversations. In *Proceedings of the* 29^{th} *IEEE Symposium on Security and Privacy*. 35–49.

Wright, C. V., Ballard, L., Monrose, F., and Masson, G. M. 2007. Language Identification of Encrypted VoIP Traffic: *Alejandra y Roberto or Alice and Bob?* In *Proceedings of the 16th Annual USENIX Security Symposium*. Boston, MA, 43–54.

Zhang, L., Wang, T., and Cuperman, V. 1997. A CELP variable rate speech codec with low average rate. In *Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 2. 735–738.

Zimmerman, P. 2008. The Zfone Project. `http://zfoneproject.com/faq.html`.

A. COMMON ENGLISH PHONEMES

	Symbol	Example Word
Stops:	b	bee
	d	day
	g	gay
	p	pea
	t	tea
	k	key
	dx	muddy, dirty
	q	bat
Affricates:	jh	joke
	ch	choke
Fricatives:	s	sea
	sh	she
	z	zone
	zh	azure
	f	fin
	th	thin
	v	van
	dh	then
Nasals:	m	mom
	n	noon
	ng	sing
	em	bottom
	en	button
	eng	washington
	nx	winner

	Symbol	Example Word
Semivowels:	l	lay
	r	ray
	w	way
	y	yacht
	hh	hay
	hv	ahead
	el	bottle
Vowels:	iy	beet
	ih	bit
	eh	bet
	ey	bait
	ae	bat
	aa	bott
	aw	bout
	ay	bite
	ah	but
	ao	bought
	oy	boy
	ow	boat
	uh	book
	uw	boot
	ux	toot
	er	bird
	ax	about
	ix	debit
	axr	butter
	ax-h	suspect

Table I. List of phoneme symbols used in the TIMIT corpus.

B. PHRASE SPOTTING PERFORMANCE ON SELECTED PHRASES

Phrase	Area
Young children should avoid exposure to contagious diseases.	0.98
Even a simple vocabulary contains symbols.	0.96
Military personnel are expected to obey government orders.	0.95
Cory and Trish played tag with beach balls for hours.	0.92
Youngsters love common candy as treats.	0.92
Weather-proof galoshes are very useful in Seattle.	0.89
Don't ask me to carry an oily rag like that.	0.89
Ralph controlled the stopwatch from the bleachers.	0.00
The best way to learn is to solve extra problems.	0.00
The bungalow was pleasantly situated near the shore.	0.00
The emblem depicts the acropolis all aglow.	0.00
The fish began to leap frantically on the surface of the small lake.	0.00
The morning dew on the spider web glistened in the sun.	0.00
The sound of Jennifer's bugle scared the antelope.	0.00

Table II. An examination of the performance of our algorithm on the eight best and worst phrases in our experiment. Phrases are ranked by the area under the Recall/Precision curves described in Section 4. Phrases that are easier to find have area close to 1.0.