Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations

Charles V. Wright   Lucas Ballard   Scott E. Coull   Fabian Monrose   Gerald M. Masson

Johns Hopkins University
Department of Computer Science
Baltimore, MD USA 21218
{*cvwright,lucas,coulls,fabian,masson*}*@jhu.edu*

Abstract

Despite the rapid adoption of Voice over IP (VoIP), 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 when the audio is encoded using variable bit rate codecs, the lengths of encrypted VoIP packets can be used to identify the phrases spoken within a call. Our results indicate that a passive observer can identify phrases from a standard speech corpus 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.

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 daily communications, researchers are continually exploring ways to improve both the efficiency and security of this new communication technology. Unfortunately, while it is well understood that VoIP packets must be encrypted to ensure confidentiality [19], it has been shown that simply encrypting packets may not be sufficient from a privacy standpoint. For instance, we recently showed that 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, it is possible to determine the language spoken in the encrypted conversation [41].

As surprising as these findings may be, one 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 work we show that the information leaked from the combination of using VBR and length preserving encryption is indeed far worse than previously thought. 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 rendering the encryption ineffective.

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

978-0-7695-3168-7 /08 $25.00 © 2008 IEEE
DOI 10.1109/SP.2008.21

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* [7] 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. In this work 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 the techniques in [41] 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 recordings of the speaker's voice nor even a single utterance of the target phrase, can identify instances of the phrase with average accuracy of 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. 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.

The remainder of the paper is organized as follows. In Section 2 we overview how VBR encoding works in VoIP and provide evidence of why we are able to infer phonemes from packet lengths. In Section 3 we discuss the requisite background for understanding profile HMMs, and how our search algorithm works. Section 4 presents our experimental methodology and results, including an analysis of how one might thwart our attack. We review related work in Section 5 and conclude in Section 6.

2 Background

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) [24], the Extensible Messaging and Presence Protocol (XMPP) [25], or an application-specific control channel like Skype [30]. The voice data is typically transmitted as a Real-time Transport protocol (RTP) [28] stream over UDP, carrying a version of the audio that has been compressed using a special-purpose speech codec such as GSM [11], G.728 [34], or several others.

Generally speaking, the codec takes as input the audio stream from the user, which is typically sampled at either 8000 or 16000 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 typically 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.

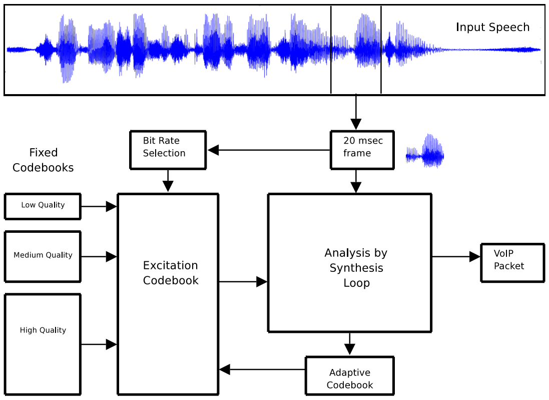

Figure 1. Basic CELP encoder

Many common voice codecs are based on a technique called code-excited linear prediction (CELP) [27] (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 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 code books used for higher-quality encodings require more bits to index, resulting in higher bit rates and therefore larger packets.

In some CELP variants, such as QCELP [9], Speex's [35] variable bit rate mode, or the approach advocated by Zhang et al. [42], 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. In a two-way call, each participant is idle roughly 63% of the time [4], so the savings may be substantial. Unfortunately, this approach can also cause substantial leakage of information in encrypted VoIP calls because, in the standard specification for Secure RTP (SRTP) [2], 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 spoken English, most in the speech community put the number between 40 and 60.

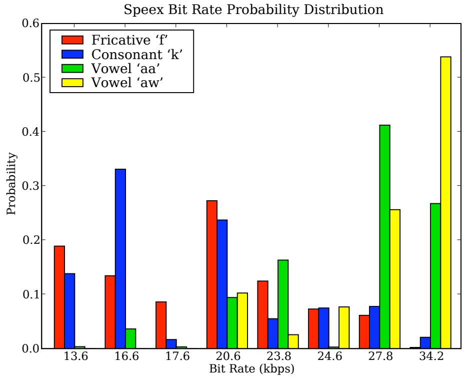

Figure 2. Distribution of bit rates used to encode four phonemes with Speex

To demonstrate the relationship between bit rate and phonemes, we encoded several recordings from the TIMIT [10] corpus of phonetically-rich English speech using Speex in wideband variable bit rate mode, and observed the bit rate used to encode each phoneme. The probabilities 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 (namely, 16.6,

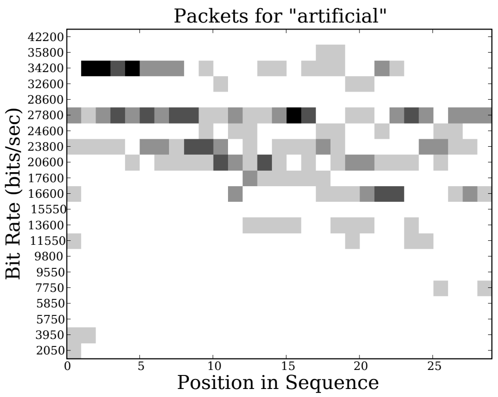

Figure 3. Packets for “artificial”

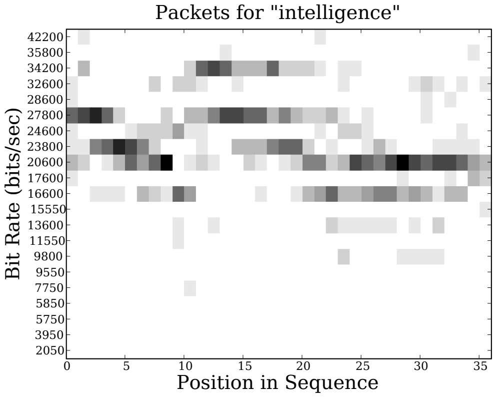

Figure 4. Packets for “intelligence”

27.8, and 34.2 kbps), can be used to distinguish *aa* from *aw* and *f* from *k*.

In fact, it is these differences in bit rate for the phonemes that make recognizing words and phrases in encrypted traffic possible. To 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 [10]. We represent the packets for each word visually in Figures 3 and 4 as a data image—a grid with 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 (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.

In the next section we detail how an eavesdropper who knows the phonetic transcription of her target phrase can compute the expected sequence of packet sizes that will be transmitted when a VoIP caller speaks the phrase. We also discuss how she can use this sequence to recognize the phrase when is spoken in a conversation.

3 Spotting Phrases with Profile HMMs

Our goal in this work is to recognize spoken words or 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. However, for ease of exposition, we begin the discussion of our machine learning techniques by first addressing a much easier scenario, where 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 no recordings of the words in the phrase she wishes to detect.

3.1 How to recognize a previously seen word or phrase

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 based on hidden Markov models [20] have proven to be also be extremely useful [40, 7]—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 20 different amino acids; in wideband mode, the Speex codec produces 21 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 [8], which were originally developed for performing multiple sequence alignment of protein families and for searching protein databases [16], to the task of finding words and phrases in encrypted VoIP. 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 [37] on the stream of packets to find subsequences of packets that match the profile. We elaborate on each of these steps below.

Building a Profile HMM A profile HMM [7] (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 so as to make the Match states the most likely state in each position.

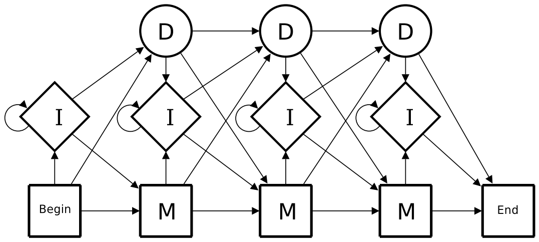

Figure 5. Profile HMM [7]

Given an initial model and a set of example sequences of packets for the target phrase, there is a well-known Expectation-Maximization [5] algorithm due to Baum and Welch [3] that uses dynamic programming 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 [15] 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 [38] to the re-

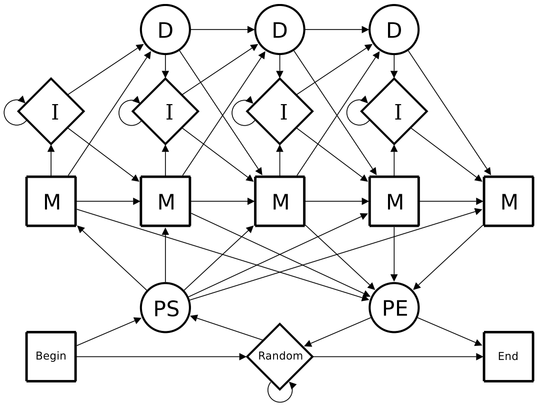

Figure 6. Search HMM [7]

sulting 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 [7].

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 5 new states to the standard profile HMM to create a search HMM (Figure 6). The most important new state is the *Random* state, shown in Figure 6 as a diamond because it, like the Insert states, 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 [37] 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

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

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.

3.2 Recognizing phrases without example utterances

In the previous section, we 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 [12]), 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, profile-like 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 [14], 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.

Building phrase models from phonemes One straightforward method for building our word and phrase models from phonemes would be to train a profile HMM for the packets produced by each phoneme, and then concatenate phoneme models in the proper order to construct word HMMs. Phrase HMMs could be similarly constructed by concatenating word HMMs. The main shortcoming of this technique is that words often have several different possible pronunciations. These differences could be attributed to variation between dialects or between individual speakers, or because of the context of the surrounding words.

Instead, to build our models, we 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. This novel approach affords us great flexibility in finding an essentially unlimited number of phrases. 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.

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.

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. So, 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 sufficient 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. So, in our example, if we have no examples 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

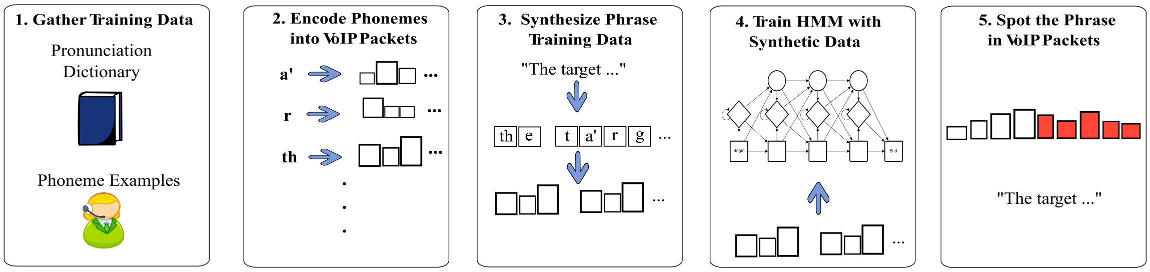
1. Gather Training Data
Pronunciation Dictionary
Phoneme Examples
2. Encode Phonemes into VoIP Packets
a'
r
th
3. Synthesize Phrase Training Data
"The target ..."
th
e
t
a'
r
g
4. Train HMM with Synthetic Data
Begin
End
5. Spot the Phrase in VoIP Packets
"The target ..."

Figure 7. Overview of training and detection process

to use a better phonetic model, using triphones, 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. For instance, if 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".

Putting it all together 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.

4 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 in an encrypted packet stream. In our experiments, we use audio recordings from the TIMIT continuous speech corpus [10], 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, denoting 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.

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. We simulate VoIP conversations for each of the speakers in the TIMIT test set by taking two copies of each of the speaker's sentences, and concatenating all of them in a random order. 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 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 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 observed during training . Explicitly, for each phrase p, let σ_p be the average log odds score 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 (see, for example Figures 8–9).

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 our performance over a range of thresholds to study the impact of the pronunciation dictionary and of noise in the audio channel on our ability to spot phrases. Then, we assume the viewpoint of an attacker and empirically estimate a specific threshold for each phrase. Finally, we discuss strategies for mitigating the information leakage that enables the attack.

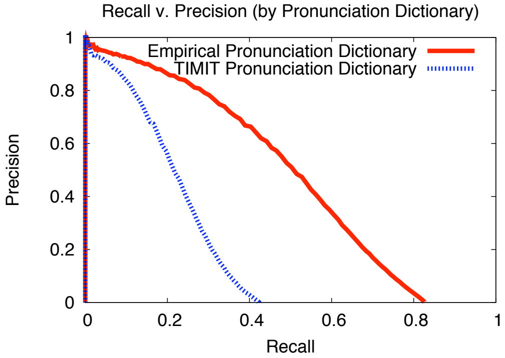

Figure 8. Comparing the performance of pronunciation dictionaries

The Importance of Accurate Pronunciations In order to build a model for a phrase, we first must know the phonemes that comprise the phrase. Although TIMIT includes a primitive pronunciation dictionary, with pronunciations given 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 realistic 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 fashion from the CALLHOME telephone speech corpus [14]. We compare the accuracy of our search HMM using these two pronunciation dictionaries and present the results 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, we are able to find, on average, 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 our the empirically derived pronunciation dictionary.

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., [33, 17, 13]).

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 for a human to determine what is being said. Figure 9 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 identify an alarming number of phrases.

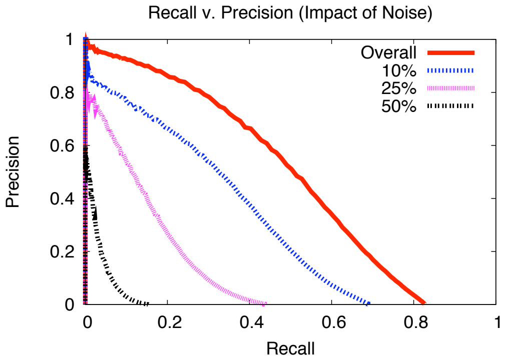

Figure 9. Results with noisy data

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 a challenging 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 earlier, 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 $t_p = \delta_p \times \sigma_p$, where δ_p 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_p that maximizes quality.

While we could define the "quality" at threshold t as either recall_t or $\mathsf{precision}_t$, 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: $TP_t - FP_t$.

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 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 for phrase p at threshold t_p.

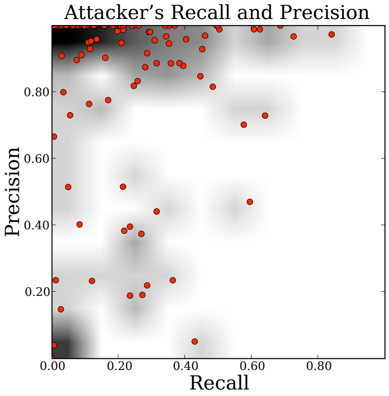

Figure 10. Attacker's accuracy

Setting k to be $1/4$ of our testing set, this technique achieves mean recall and precision rates of (.32, .75). Given that our original averages were (.50, .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 10. Each of the points denotes 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!

Analysis of Results While our approach performs well on average, there are also several phrases that we can find with great accuracy. Figure 11 shows precision and recall for four interesting phrases. We exhibited the highest accuracy when searching for the phrase *"Young children should avoid exposure to contagious diseases."*. For this phrase, our technique achieves a precision of 1.0 and a recall of .99. We also perform well on the phrase *"The fog prevented them from arriving on time."*, achieving .84 precision and .72 recall. These results illustrate the success of our technique in identifying words and phrases we have never seen before, as neither occurs in our training set. Also noteworthy are phrases *"She had your dark suit in greasy wash water all year."* and *"Don't ask me to carry an oily rag like that."* which were the only two phrases spoken by every user in the TIMIT database. We achieve precision/recall scores of (.90/.82) and (.92/.81), respectively.

Naturally, given that we are searching for phrases in encrypted audio traffic, identifying each phrase exactly can be extremely challenging. Sometimes, when our search model misses an instance of the target phrase, it only misses one or two of the words at the beginning or at the end of the phrase. Because our very strict definition of a true positive excludes such hits, it may underestimate the practical performance of our technique.

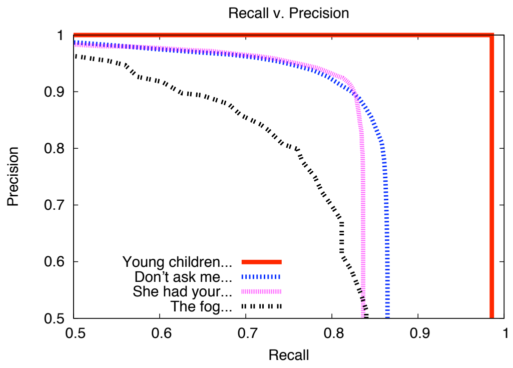

Figure 11. Performance on selected phrases

When we designate hits that contain at least $n - 2$ of the n words in the phrase as true positives, the algorithm's recall and precision improve to .55 and .53, respectively. Compared to our original, stricter classification, this represents improvement of 9% in recall and 4% in precision.

To identify other causes of the differences in accuracy between phrases, we examined several features of the phrases, including their length, phonetic composition, and the distribution of packet sizes for the words and phonemes in the phrase. Interestingly, we found no statistically significant correlation between recognition accuracy and the frequency of any of the phonemes. Given that TIMIT was designed as a phonetically rich corpus, we believe this shows that our technique is robust and flexible enough to handle the vast majority of words in spoken English.

According to our analysis, the most important factors in determining our ability to recognize a given phrase in the TIMIT data are: (1) the length of the phrase in packets, and (2) the individual speakers who spoke the phrase. Short phrases are difficult to spot reliably because it is much more likely that short patterns of packets will occur randomly in other speech. Therefore, as the length of the phrase increases, the number of false positives from the search HMM decreases and the detector's precision increases. Our detector achieves its best results on phrases that are at least 3 seconds in length.

The most important factor in determining our detector's recall was not one which we initially anticipated. It appears that there are some speakers in the dataset whom we can recognize with great accuracy, and some with whom we have more difficulty. Our technique for synthesizing training data for the profile HMM does not seem to accurately predict the way everyone speaks. To see the variability in our performance across the 168 speakers in the test set, we computed the attacker's true positive rate for each speaker s in the test set, as the fraction of utterances from s that our algorithm detects. The median true positive rate for speakers is 63%, and for about 20% of the speakers the true positive rate is below 50%. When a phrase happens to be spoken by several users for whom our synthesis techniques do not work well, our true positive rate for the phrase suffers as well. This impacts both precision and recall, because the true positive rate factors strongly in both measures.

Techniques for Mitigation One way to prevent word spotting would be to pad packets to a common length, or at least to coarser granularity. 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 12. The use of padding is quite encouraging as a mitigation technique, as it greatly reduced the overall accuracy of the search algorithm. When padding to multiples of 128 bits, we achieve only 0.15 recall at 0.16 precision. Increasing padding so that packets are multiples of 256 bits gives a recall of .04 at .04 precision. That said, padding to 128, 256, and 512 bit blocks results in overheads of 8.81%, 16.5%, and 30.82%, respectively. These bandwidth estimates are likely lower than the overhead incurred in practice, because as Chu notes [4], in a two-way call each participant is idle 63% of the time, which would allow the transmission of many smaller packets. However, our testing is comprised of continuous speech, and so the smaller packets that indicate silence are less prevalent.

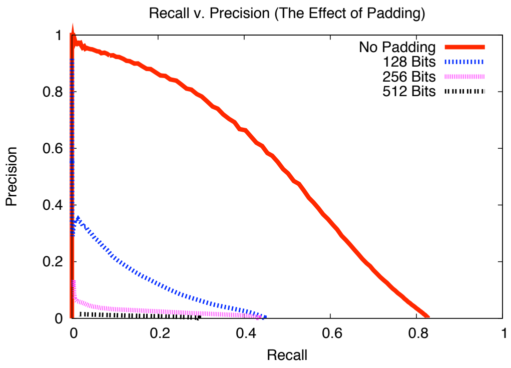

Figure 12. Robustness to padding

5 Related Work

In 1982, Simmons and Holdridge [29] 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. [39] proposed a method of tracking VoIP calls across anonymizing networks, like ToR [6], through the use of packet timing as a watermark. Verscheure et al. [36] 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. [18] and Aggarwal et al. [1] 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 [41].

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. [32] 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 responses. More recently, Saponas et al. [26] 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. Packet inter-arrival times have been used to infer keystrokes within encrypted SSH sessions [31].

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., [22, 40]), or the composition of phonemes into words (e.g., [21, 23]). 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, the performance of our approach is not significantly worse than that of early wordspotting methods in speech.

6 Conclusion

Previous work has shown that combining VBR compression with length-preserving encryption leaks information about VoIP conversations [41]. In this paper, we show that this information leakage is far worse than originally thought. Our results indicate that a profile hidden Markov model trained using speaker- and phrase-independent data can detect the presence of some phrases within encrypted VoIP calls with recall and precision exceeding 90%. On average, our method achieves recall of 50% and precision of 51% for a wide variety phonetically rich phrases spoken by a diverse collection of speakers. Moreover, we examine the impact of noise, dictionary size, and word variation on the performance of our techniques.

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 [2]. 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 focuses on the development of padding techniques that provide an appropriate balance between efficiency and security.

References

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

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

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

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

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

[6] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the* 13^{th} *USENIX Security Symposium*, pages 303–320, August 2004.

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

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

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

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

[11] Global System for Mobile communications. `http://www.gsmworld.com/index.shtml`.

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

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

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

[15] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

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

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

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

[19] N. Provos. Voice over misconfigured internet telephones. `http://vomit.xtdnet.nl`.

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

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

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

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

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

[25] P. Saint-Andre. Extensible messaging and presence protocol (XMPP): Core. RFC 3920.

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

[27] M. R. Schroeder and B. S. Atal. 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*, volume 10, pages 937–940, April 1985.

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

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

[30] Skype. `http://www.skype.com`.

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

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

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

[34] I. T. Union. Coding of speech at 16kbit/s using low-delay code excited linear prediction, September 1992.

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

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

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

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

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

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

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

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