

Whitepaper

Practical Attacks against
Encrypted VoIP Communications

September 2013



Prepared by: Dominic Chell, Shaun Colley
E-Mail: research [at] mdsec.co.uk

MDSec Consulting Ltd, @MDSecLabs

Contents

1.	Introduction	3
2.	Previous Work	4
3.	VoIP Background Information.....	5
3.1.	Control Channel	5
3.2.	Data Channel.....	6
3.3.	Codecs.....	6
	Variable Bitrate Codecs	7
4.	Natural Language Processing.....	9
4.1.	Hidden Markov Models	9
	Best Path.....	11
	Probability of a Sequence	11
	Training	11
	Profile Hidden Markov Models.....	12
4.2.	Dynamic Time Warping	13
5.	Side Channel Attacks	16
5.1.	Profile Hidden Markov Models	16
	Collecting training data	18
	Searching and Scoring	19
5.2.	Dynamic Time Warping	20
	Collecting training data	20
	Speaker Independence.....	21
	Scoring	21
6.	Conclusions.....	22

1. Introduction

VoIP has become a popular replacement for traditional copper-wire telephone systems as businesses look to take advantage of the bandwidth efficiency and low costs that are associated with the technology. Indeed, in March 2013 Point Topic recorded the combined total of global VoIP subscribers to be 155.2 million¹. With such a vast subscriber base in both consumer and corporate markets, in the interests of privacy it is imperative that communications are secured.

The privacy associated with popular VoIP software is increasingly a concern, not only for individuals but also for corporations whose data may be discussed in VoIP phone calls. Indeed, this has come under greater scrutiny in light of accusations of wiretapping and other capabilities against encrypted VoIP traffic, such as the PRISM and BULLRUN programmes allegedly operated by the NSA and GCHQ².

Like with many transports, it is generally accepted that encryption should be used to provide end-to-end security of communications. While there is extensive work covering the security of VoIP control channels and identifying implementation flaws, little work that assesses the security of VoIP data streams has been published.

This whitepaper detail demonstrable methods of retrieving information from spoken conversations conducted over encrypted VoIP data streams. This is followed with a discussion of the possible ramifications this may have on the privacy and confidentiality of user data in real world scenarios.

¹ <http://point-topic.com/free-analysis/global-voip-subscriber-numbers-q1-2013/>

² <http://www.theguardian.com/world/2013/sep/05/nsa-gchq-encryption-codes-security>

2. Previous Work

There is very little previous work from the security community that has been published in this area. However, several notable academic papers discuss traffic analysis of VoIP communications in detail. In particular, the following publications are relevant:

- Language Identification of Encrypted VoIP Traffic
Charles V. Wright Lucas Ballard Fabian Monroe Gerald M. Masson
<http://www.cs.jhu.edu/~cwright/voip-vbr.pdf>
- Uncovering Spoken Phrases in Encrypted Voice over IP Communications
Charles V. Wright, Lucas Ballard, Scott E. Coull, Fabian Monroe, Gerald M. Masson
<http://www.cs.jhu.edu/~cwright/voip-vbr.pdf>
- Uncovering Spoken Phrases in Encrypted VoIP Conversations
Goran Doychev, Dominik Feld, Jonas Eckhardt, Stephan Neumann
<http://www.infsec.cs.uni-saarland.de/teaching/WS08/Seminar/reports/yes-we-can.pdf>
- Analysis of information leakage from encrypted Skype conversations
Benoît Dupasquier, Stefan Burschka, Kieran McLaughlin, Sakir Sezer
<http://link.springer.com/article/10.1007%2Fs10207-010-0111-4>

3. VoIP Background Information

Within this section, we provide the reader with a brief overview of the fundamentals of VoIP communications and the essential background information specific to understanding our attack.

Similar to traditional digital telephony, VoIP communications involve signalling, session initialisation and setup as well as encoding of the voice signal. VoIP communications can typically be separated in to two separate channels that perform these actions; the control channel and the data channel.

3.1. Control Channel

The control channel operates at the application-layer and performs the call setup, termination and other essential aspects of the call. To achieve this, a signalling protocol is used with popular open implementations including: the Session Initiation Protocol; the Extensible Messaging and Presence Protocol; and H.323; as well as closed, application dependent protocols such as Skype.

Control channel communications will exchange sensitive call data such as details on the source and destination endpoints and can be used for modifying existing calls. As such, many signalling implementations will support encryption to protect the data exchange; an example of this is SIPS which adds Transport Layer Security (TLS) to the SIP protocol. The control channel is typically performed over TCP and is used to establish a direct UDP data channel for voice traffic to be transferred. It is this data communication channel that is the primary focus of this research, as opposed to the signalling data.

3.2. Data Channel

Voice data is digitally encoded, and in some cases compressed, before being sent over the network via UDP in the data channel. The voice data will typically be transmitted using a transport protocol such as Real-time Transport Protocol (RTP)³ or a similar derivative.

Due to the often sensitive nature of the content being communicated across the data channel, it is commonplace for VoIP implementations to encrypt the data flow to provide confidentiality. Perhaps the most common way this is achieved is using the Secure Real-time Transport Protocol (SRTP)⁴.

SRTP provides encryption and authentication of the encoded RTP stream, however it does not apply padding and thus preserves the original RTP payload size. Indeed, the RFC specifically states:

"None of the pre-defined encryption transforms uses any padding; for these, the RTP and SRTP payload sizes match exactly."

As a consequence, in some scenarios this leads to information leakage that can be used to deduce call content, as discussed in greater detail later.

3.3. Codecs

Codecs are used to convert the analogue voice signal into a digitally encoded and compressed representation. In VoIP, there will always be a trade-off between bandwidth limitations and voice quality; it is the codec that determines how to strike a balance between the two.

³ <http://www.ietf.org/rfc/rfc3550.txt>

⁴ <http://www.ietf.org/rfc/rfc3711.txt>

Perhaps the most widely used technique for speech analysis is the Code-Excited Linear Prediction (CELP)⁵ algorithm.

CELP encoders work by trying all possible bit combinations in the codebook and selecting the one that is the closest match to the original audio, essentially performing a brute-force. In some CELP implementations and similar encoder variations, the encoder determines a varying bit rate for each packet in the encoded stream with the aim of achieving a higher quality of audio without a significant increase in bandwidth.

Variable Bitrate Codecs

When encoding a speech signal, the bit rate is the number of bits over time required to encode speech, typically this is measured in either bits per second or kilobits per second. Variable bit-rate (VBR) implementations allow the codec to dynamically modify the bit-rate of the transmitted stream. In codecs such as Speex⁶ when used in VBR mode, the codec will encode sounds at different bit rates. For example, Speex will encode fricative consonants⁷ at a lower bit rate than vowels.

Consider the following graph which shows the packet lengths of a sentences containing a number of fricatives, over time:

⁵ http://en.wikipedia.org/wiki/Code-excited_linear_prediction

⁶ <http://www.speex.org/>

⁷ http://en.wikipedia.org/wiki/Fricative_consonant

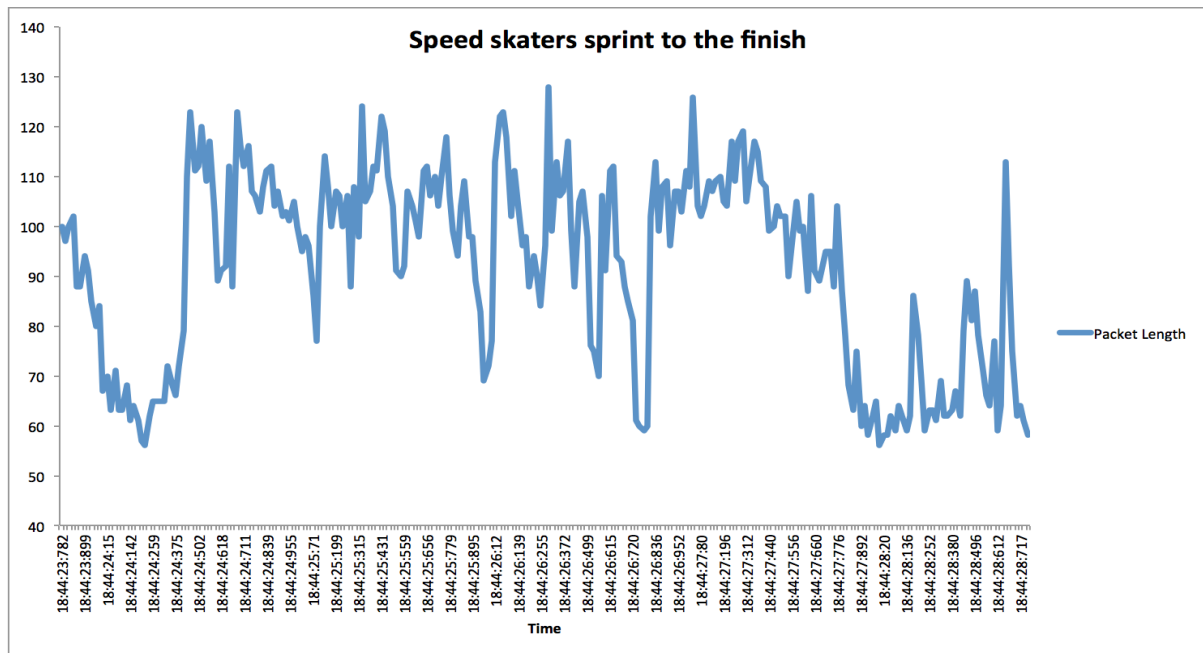


Figure 1: Packet lengths over time

It can be seen from the graph, that there are a number of troughs. These can be roughly mapped to the fricatives in the sentence.

The advantage of VBR codecs is primarily that it produces a significantly better quality-to-bandwidth ratio when compared with a constant bit rate codec and so poses an attractive choice for VoIP; especially as bandwidth may not be guaranteed.

4. Natural Language Processing

The techniques we use in our work and demonstrations to elucidate sensitive information from encrypted VoIP streams are borrowed from the Natural Language Processing (NLP) and bioinformatics communities.

The two main techniques we use in our attacks are profile Hidden Markov Models (HMM)⁸ and Dynamic Time Warping (DTW). Thanks to their ability to perform types of sequence and pattern matching, both of these methods have found extensive use in NLP (i.e. DTW and HMM for speech recognition) and bioinformatics (i.e. HMM for protein sequence alignment).

We will now cover some background on both of these techniques in order to explore their relevance in VoIP traffic analysis attacks.

4.1. Hidden Markov Models

Hidden Markov Models (HMM) are a type of statistical model that assign probabilities to sequences of symbols. A HMM can be thought of as a model that generates sequences by following a series of steps.

A Hidden Markov Model consists of a number of finite states. It always begins in the Begin state (B), and ends in the End state (E). In order to move from state B to state E the model moves from state to state, randomly, but according to a *transition* distribution. For example, if a transition from state T to state U happens, this happens according to T's transition distribution. It's worth noting that since these transitions are Markov processes, each transition happens independently of all other choices previous to that transition; the step only depends on what state the HMM is currently in.

⁸ http://en.wikipedia.org/wiki/Hidden_Markov_model

When a transition occurs, and the HMM finds itself in a *silent* state, the model just decides where to transition to next, according to the state's transition distribution. The state is silent in the sense that no symbol is emitted. However, if the state is not silent, the model picks an output symbol according to the state's emission distribution, outputs this symbol, and then carries on with transitioning from state to state. As the model continues to move between states, these emitted symbols constitute the HMM's outputted sequence, until state E is reached, at which point the process terminates. The B and E states are silent states. Consider the diagram, which illustrates a hypothetical state path.

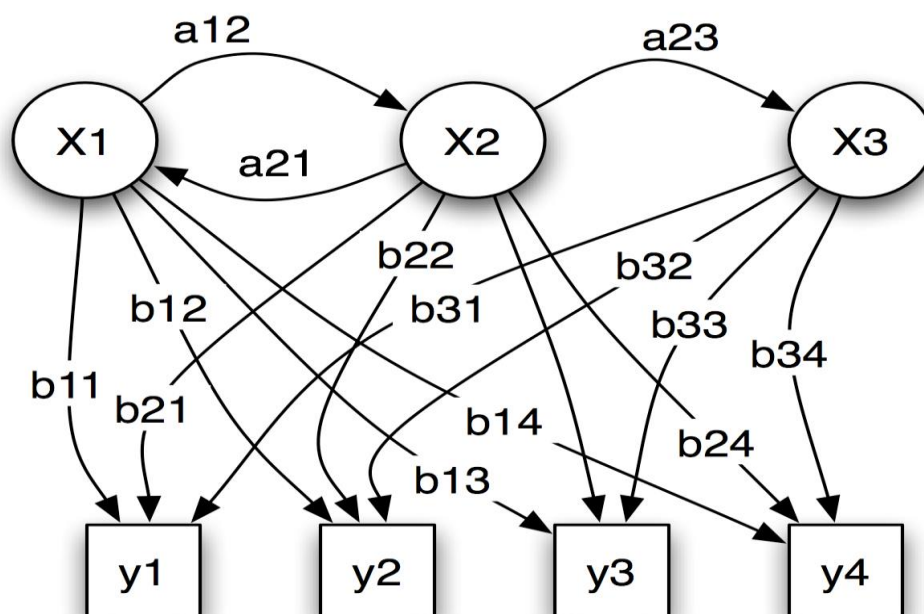


Figure 2: Example State Path
(<http://commons.wikimedia.org/wiki/File:HiddenMarkovModel.png>)

Best Path

Although there can be a great number of possible state paths that the HMM can take from state B to E, there is always a best path for each possible output sequence. It follows that since this is the best path, it is also the most likely path. The Viterbi algorithm can be used to discover the most probable path for a given observation sequence. The Viterbi⁹ algorithm uses dynamic programming techniques, and although a description is beyond the scope of this document, many explanations and implementations of Viterbi are available on the Internet.

Probability of a Sequence

In addition to being able to find the best path for an observation sequence, it is also useful to be able to compute the probability of a model outputting an observation sequence; the Forward and Backward¹⁰ algorithms are useful for this purpose. Having the ability to determine the probability of a model producing a specific output sequence has particularly useful applications, and has seen widespread use in bioinformatics (i.e. protein sequence alignment) and Natural Language Processing, such as for speech recognition. One of our attacks, discussed later, will rely on all three of the algorithms mentioned thus far; Viterbi, Forward and Backward.

Training

The real usefulness of HMMs becomes apparent when considering that Hidden Markov Models can be *trained* according to a collection of output sequences.

The Baum-Welch algorithm¹¹ is commonly used to estimate (making use of the Forward and Backward algorithms) the

⁹ http://en.wikipedia.org/wiki/Viterbi_algorithm

¹⁰ http://en.wikipedia.org/wiki/Forward%E2%80%93backward_algorithm

¹¹ http://en.wikipedia.org/wiki/Baum%E2%80%93Welch_algorithm

emission and transition probabilities of a model, assuming it previously output a particular set of observation sequences.

Thus, we can essentially build a HMM from a set of training data. Following this, we could then “ask” the model using Viterbi or Forward/Backward what the probability is of an arbitrary sequence having been produced by the model. This allows us to train a HMM with a collection of data, and then use the model to recognise similar sequences to the training data.

This forms the very basis of using HMMs for the many types of pattern and speech recognition. Of course, in the context of say, speech recognition, “sequences of symbols” would perhaps be sequences of signal amplitudes, and in the context of protein sequence alignment, the possible output symbols would be the four amino acids.

Profile Hidden Markov Models

Profile Hidden Markov Models are a type of HMM. The most notable addition to standard HMM topologies are the addition of *insert* and *delete* states. These two states allow HMMs to recognise sequences that have additions or insertions. For example, consider the following hypothetical sequence, which a HMM has been trained to recognise:

A B C D

With the presence of insert and delete states, the model is still likely to recognise the following sequences, which have an insertion and deletion, respectively:

Insertion

A B **X** C D

Deletion

A B D

Profile HMMs are particularly useful for application in traffic analysis as outputs of audio codecs and transmission as IP packets will seldom be identical even for utterances of the same phrase even by the same speaker. For this reason, we need our models to be more “forgiving”, since IP traffic is very unlikely to be identical even for very similar audio inputs.

4.2. Dynamic Time Warping

Dynamic Time Warping is an algorithm for measuring the similarity between two sequences, which may vary in time or speed. DTW has seen widespread use in speech recognition, speaker recognition, signature recognition and other video, audio and graphical applications.

Although DTW is an older and somewhat simpler technique than HMMs that has largely been replaced by HMMs, DTW is still of interest to us in our traffic analysis attack because it takes into account the temporal element that network traffic intrinsically has. A stream of network packets or datagrams, in essence, constitutes a time series.

Furthermore, speech is also a time-dependent process, which is what these attacks are focused on. A speaker may utter a phrase in a similar manner to another person with a similar accent, but they may utter the phrase faster or slower. DTW was in fact first used to recognise similar utterances which were spoken at different speeds.

To illustrate this with an example, consider the two sequences of integers:

0 0 0 4 7 14 26 23 8 3 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 6 13 25 24 9 4 2 0 0 0 0 0

If we simply compared these sequences “component-wise” the two appear to be very different. However, if we compare their characteristics, they have some similarities; the sequences are both 8 integers in length, and they both have a “peak” at 25-26. Simply comparing these sequences from their entry points disregards features of the sequences that we think of as “shape” (i.e. if plotted).

In the context of speech recognition applications, one of the sequences is the sequence “to be tested”, such as an incoming voice signal, and the other sequence is a prototypical sequence considered to be typically produced by some process. In speech recognition, this would be a typical utterance of the phrase in question; generally known as a “template”.

The two sequences can be arranged perpendicular to one another on adjacent sides of a grid, with the input sequence on the bottom, and the template sequence up the vertical side. Consider the diagram below.

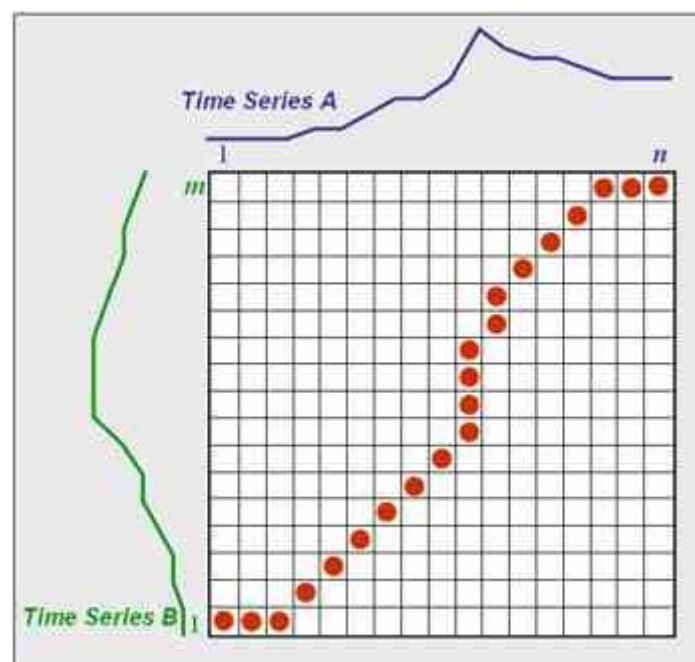


Figure 3: DTW Time Series (<http://cst.tu-plovdiv.bg/bi/DTWimpute/DTWalgorithm.html>)

Inside each of the cells we then place a distance measure comparing the corresponding elements of the two sequences. The best match between these sequences is then found by finding a path through the grid that minimises the total distance between them. From this, the overall distance between the two sequences is calculated, giving an overall distance metric. This may be known as the DTW distance.

Accordingly, this metric yields how similar the two sequences are.

5. Side Channel Attacks

With the necessary background aptly covered, we now describe the two attacks that will be demonstrated at HackinTheBox (2013, Kuala Lumpur). The associated proof of concepts can be found on the MDSec website following the conference (<http://www.mdsec.co.uk>).

We describe here our traffic analysis attack using profile Hidden Markov Models for traffic analysis, using Skype as the case study. Later, we also describe an attack that uses Dynamic Time Warping, with the same aim of “spotting” sentences and phrases in Skype conversations.

5.1. Profile Hidden Markov Models

Skype uses the Opus codec¹² in VBR mode and as previously noted, spoken phonemes are generally reflected in the packet lengths when in VBR mode. Since Skype uses AES encryption in ICTR mode (“integer counter” mode), the resulting packets are not padded up to specific size boundaries.

Consequently, this means that similar utterances generally result in similar sequences of packet lengths. Consider the following graph, which represents the payload lengths vs. time plotted for three packet captures; two of the same phrase versus an utterance of a completely different phrase. All phrases were spoken by the same speaker over a Skype voice conversation. Note the following packet dumps were not collected under proper experimental conditions; the plots below simply aim to demonstrate the audio input vs. packet payload length relationship.

¹² <http://www.opus-codec.org/>

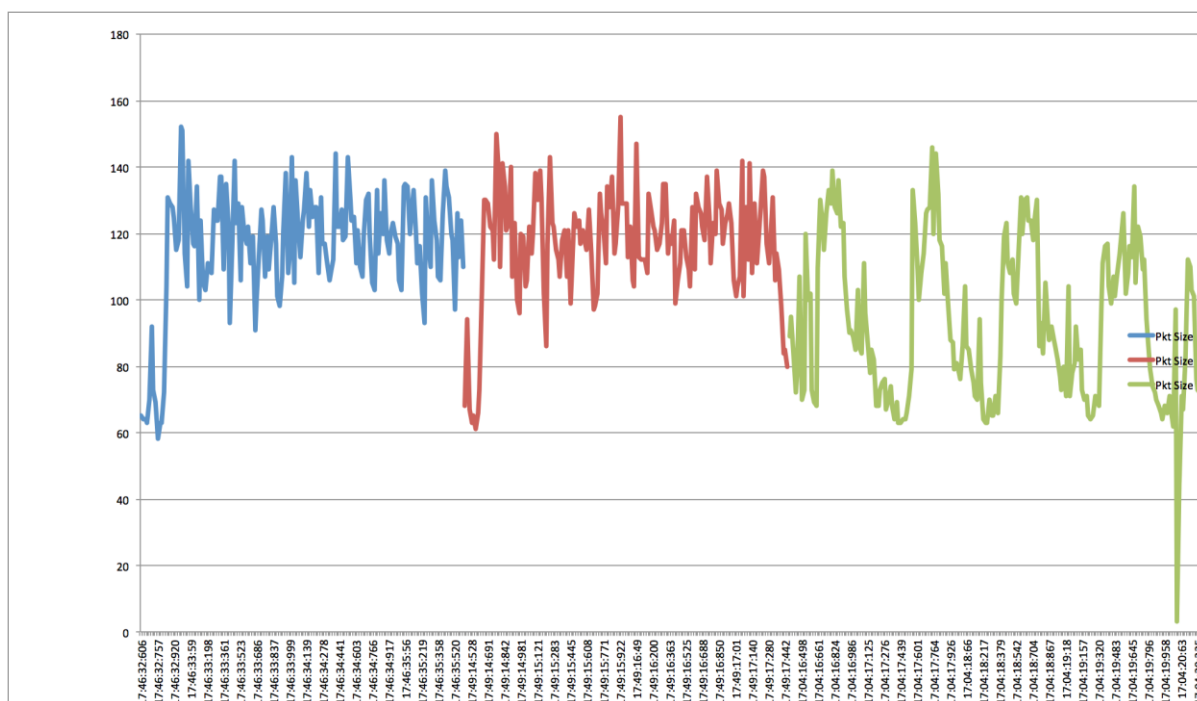


Figure 4: Packet Lengths over time. Blue and red represent the same phrase, green is a different phrase

The fact that similar utterances bear resemblance to one another represents a significant information leak; it shouldn't be possible to divulge any information about the nature of encrypted content whatsoever. This issue may be referred to as a side-channel attack, since attacks don't reveal the actual contents of encrypted conversations; instead, analytical techniques are used to deduce the information.

It should be noted, however, that similar utterances of a given phrase or sentence seldom produce the *exact* same sequence of packet lengths. There are several reasons for this; among these are accent differences between different speakers, background noise and speed at which the phrase is spoken. It is therefore not possible to spot spoken phrases via a substring matching method, since even utterances by the same speaker will not yield the exact same sequence of packet lengths.

One solution is the use of the profile Hidden Markov Model. Such an attack, in its most basic form, can be used to spot

known phrases in Skype traffic. The attack can be summarised as follows:

- 1) Train a profile HMM for the target phrase,
- 2) Capture Skype traffic,
- 3) "Ask" the profile HMM if the test sequence is likely to be an utterance of the target phrase.

Collecting training data

Our primary requirement to build a profile HMM for a target phrase is training data; that is, many packet captures of traffic that resulted in the target phrase being spoken or played over a Skype voice chat. Our approach to this was a simple one; we first created a directory containing all samples we wished to include in the dataset, in RIFF¹³ format. We then setup a packet sniffer – tcpdump, in our case – and initiated a voice chat between two Skype accounts. This resulted in encrypted UDP traffic between the two computers, in a "peer-to-peer" fashion, i.e. directly between the two systems.

We played each of the soundtracks across the Skype session using VLC Media player¹⁴, with five second intervals of silence between each track. A BASH loop similar to the following was used:

```
for((a=0;a<400;a++)); do
/Applications/VLC.app/Contents/MacOS/VLC --no-repeat
-I rc --play-and-exit $a.rif ; echo "$a " ; sleep 5 ;
done
```

Meanwhile, tcpdump was configured to log all traffic flowing to the other test system.

```
tcpdump -w training.pcap dst <dest_ip>
```

¹³ http://en.wikipedia.org/wiki/Resource_Interchange_File_Format

¹⁴ <http://www.videolan.org>

Once all training data had been collected, sequences of UDP payload lengths were extracted via means of automated PCAP file parsing.

The resulting payload length sequences were then used to train a profile HMM using the Baum-Welch algorithm.

It should be noted that all collection of training data should be carried out in quiet environments with little background noise.

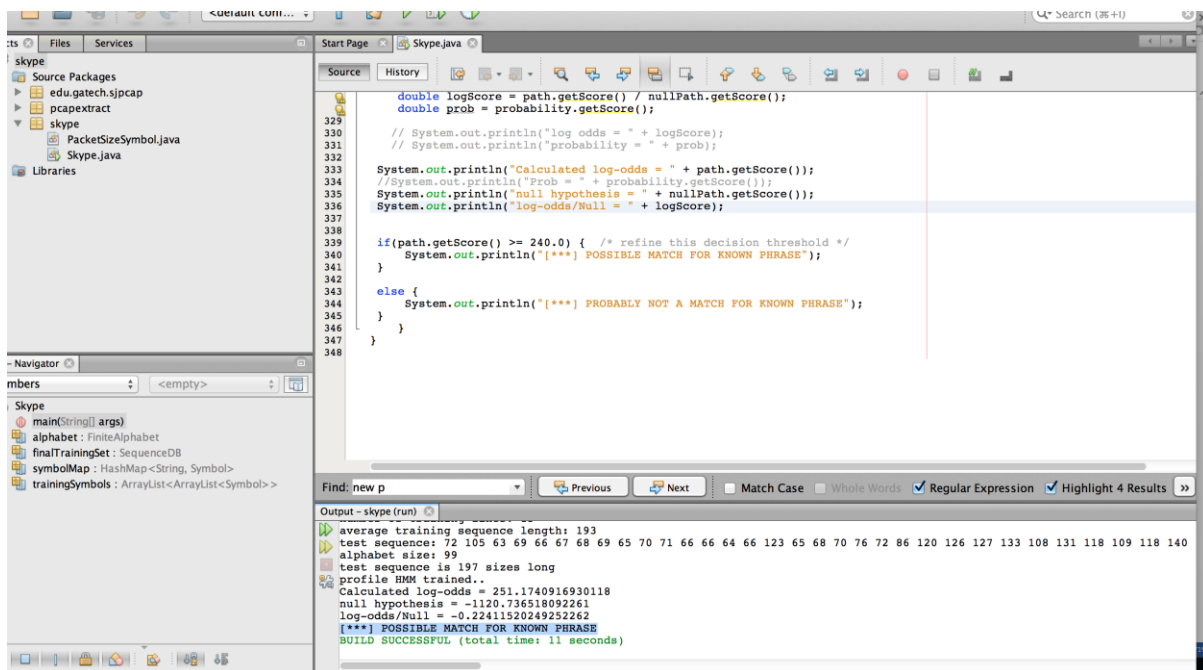
Recordings of particular sentences were selected to avoid speakers with radically different accents and timings.

Searching and Scoring

Once a viable model has been formed from sensible training data, sequences of packet lengths can be “queried” against the model; in an attempt to determine how likely it is that the traffic corresponds to an utterance of the target phrase.

A scoring threshold must be established. A log-odds (or otherwise) score above which the traffic was considered a “hit” (traffic matched the phrase) must be decided on manually. Then, accordingly, if a payload length sequence scores above this threshold, we consider it a “hit”, and if not, a “miss” is recorded.

The following screenshot demonstrates the output from our proof of concept code when provided with a PCAP file for a phrase that exists within the training data:



```

double logScore = path.getScore() / nullPath.getScore();
double prob = probability.getScore();

// System.out.println("log odds = " + logScore);
// System.out.println("probability = " + prob);

System.out.println("Calculated log-odds = " + path.getScore());
//System.out.println("Prob = " + probability.getScore());
System.out.println("null hypothesis = " + nullPath.getScore());
System.out.println("log-odds/Null = " + logScore);

if(path.getScore() >= 240.0) { /* refine this decision threshold */
    System.out.println("[***] POSSIBLE MATCH FOR KNOWN PHRASE");
}
else {
    System.out.println("[***] PROBABLY NOT A MATCH FOR KNOWN PHRASE");
}
}
}

Find: new p
Previous Next Match Case Whole Words Regular Expression Highlight 4 Results >>

Output - skype (run)
average training sequence length: 193
test sequence: 72 105 63 69 66 67 68 69 65 70 71 66 66 64 66 123 65 68 70 76 72 86 120 126 127 133 108 131 118 109 118 140
alphabet size: 99
test sequence is 197 sizes long
profile HMM trained..
Calculated log-odds = 251.1740916930118
null hypothesis = -1120.736518092261
log-odds/Null = -0.22411520249252262
[***] POSSIBLE MATCH FOR KNOWN PHRASE
BUILD SUCCESSFUL (total time: 11 seconds)

```

Figure 5: A phrase being detected in an encrypted Skype conversation

5.2. Dynamic Time Warping

This attack makes use of the Dynamic Time Warping algorithm to “spot” sentences, similarly to the previous profile HMM attack. We do this for comparison of the efficacies of the two techniques and to demonstrate two different methodologies that can be used for traffic analysis, and in particular, sentence spotting in encrypted traffic streams.

Collecting training data

As opposed to the profile HMM method, DTW does not require a large set of training data. We collect data in much the same way as in the profile HMM experiment. That is, by playing audio samples over a Skype session using a loop similar to the following:

```

for((a=0;a<400;a++)); do
/Applications/VLC.app/Contents/MacOS/VLC --no-repeat
-I rc --play-and-exit $a.rif ; echo "$a " ; sleep 5 ;
done

```

And, as before, the data is captured via tcpdump, i.e.

```
tcpdump -w training.pcap dst <dest_ip>
```

Each packet sequence was extracted from the resulting PCAP file in an automated fashion, and models were created for each using the DTW algorithm. Each utterance was the exact same recording being played over the Skype conversation.

Speaker Independence

Based on the suggestions of Benoît Dupasquier et al.¹⁵, the Kalman filter was applied to each of the training data sets to avoid the need for large amounts of training data. In this way, speaker-dependence is somewhat removed from the template models created.

Scoring

The DTW algorithm is then used to compare test data to the prepared models. This produces a DTW distance between the test packet sequence and the template models. The DTW distance is then compared to a predetermined scoring threshold and is accordingly deemed to be a probable “hit” or probable “miss” with respect to the target sentence or phrase.

¹⁵ <http://link.springer.com/article/10.1007%2Fs10207-010-0111-4>

6. Conclusions

Our research has concluded that Variable Bit Rate codecs are unsafe for sensitive VoIP transmission, when encrypted with a length preserving cipher. Indeed, the results of our research demonstrated that given sufficient training data, it is possible to deduce spoken conversations in encrypted transmissions.

Our results indicate that using Profile Hidden Markov Models analysis techniques, it is possible, in some cases, to achieve over 90% reliability in discovering spoken phrases in encrypted conversations. Additionally, it is in some cases possible to detect known phrases in conversations using or Dynamic Time Warping with over 80% reliability, using much less training data than in Profile HMM attacks.

Consequently, the use of a codec in VBR mode with an encrypted transport such as SRTP or other VoIP protocols, with its default encryption, should be avoided.

Some guidance is offered in RFC6562¹⁶ for use of VBR codecs with SRTP which suggests that RTP padding may provide a reduction in information leakage. However, ultimately in scenarios where by a high degree of confidentiality is required it is advised that a Constant Bit Rate codec is negotiated during VoIP session initiation.

¹⁶ <http://tools.ietf.org/html/rfc6562>