

Inferring Info from Encrypted Network Traffic

Kevin Z. Snow

kzsnow@cs.unc.edu

October 1, 2009

Overview

- Going to look at nothing but size and timing of packets
- Three applications covered
 - Recovering keystrokes in an SSH session
 - Identifying the protocol of a network flow
 - Recognizing spoken phrases in an encrypted VoIP call

Overview

- Going to look at nothing but size and timing of packets
- Three applications covered
 - Recovering keystrokes in an SSH session
 - Identifying the protocol of a network flow
 - Recognizing spoken phrases in an encrypted VoIP call
- We will cover the following ML topics:
 - HMM topologies (simple, profile, search)
 - Viterbi, n-Viterbi
 - Vector Quantization
 - Model surgery

Recovering SSH Keystrokes

Paper

Song, D. X., Wagner, D., and Tian, X. *Timing analysis of keystrokes and timing attacks on SSH*. USENIX Security Symposium, 2001

Goal

Recover the keys pressed during an SSH session, specifically to capture a password

What is SSH

"SSH was designed as a replacement for Telnet and other insecure remote shells, which send information, notably passwords, in plaintext, leaving them open for interception. The encryption used by SSH provides confidentiality and integrity of data over an insecure network, such as the Internet."

-Wikipedia

Recovering SSH Keystrokes

- Provides strong authentication and encryption between two hosts
- Despite this, information can still be leaked

Recovering SSH Keystrokes

- Provides strong authentication and encryption between two hosts
- Despite this, information can still be leaked
 - Packets only padded to an 8-byte boundary, adversary can therefore approximate the original data length (e.g. is the SSH password more than 7 characters?)
 - In interactive mode, each keystroke is sent in a separate packet after the key is pressed

Interactive Mode Example

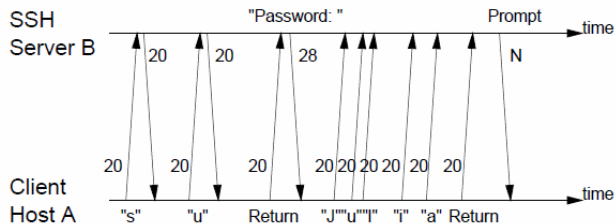
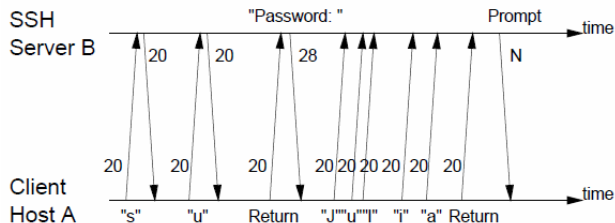


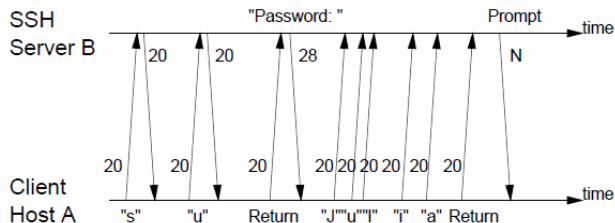
Image Credit: Dawn Song et al

Interactive Mode Example



- Signature could be built on packet size

Interactive Mode Example



- Signature could be built on packet size
- Keystroke timing may be useful
 - Follows a stable pattern, prior to this work it had been considered for use as a biometric for user authentication
 - But, does keystroke timing leak information about the keys being typed? Intuitively, yes.

Exploring Keystroke Timing Info Leakage

- Empirical experiment measuring inter-keystroke timing
 - Users typed each character pair, (k_a, k_b) , 30-40 times for 142 pairs in the experiment
 - Collected latency of each character pair

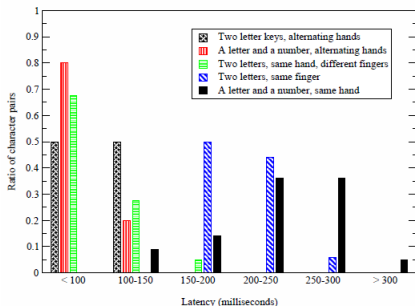
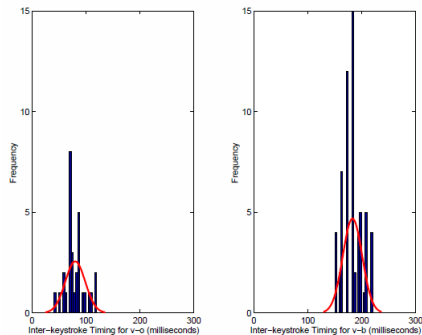
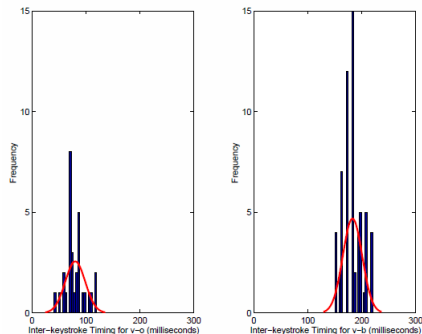


Image Credit: Dawn Song et al

Modeling Keystroke Pairs

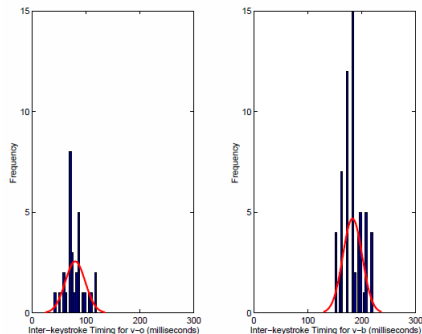


Modeling Keystroke Pairs



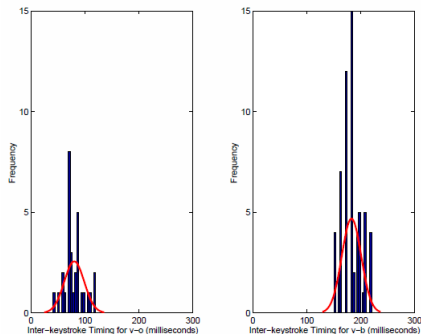
- The latency, y , between keystrokes of a character pair $q \in Q$, $Pr[y|q]$ is modeled as a univariate Gaussian distribution

Modeling Keystroke Pairs



- The latency, y , between keystrokes of a character pair $q \in Q$, $Pr[y|q]$ is modeled as a univariate Gaussian distribution
- The distributions parameters, $\{(\mu_q, \sigma_q)\}_{q \in Q}$, are derived from the maximum likelihood estimation...

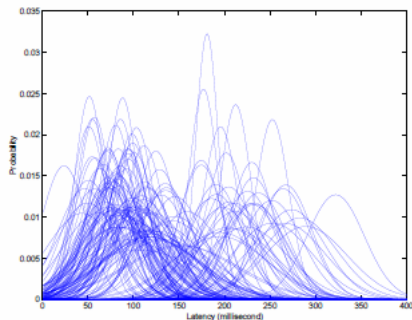
Modeling Keystroke Pairs



- The latency, y , between keystrokes of a character pair $q \in Q$, $Pr[y|q]$ is modeled as a univariate Gaussian distribution
- The distributions parameters, $\{(\mu_q, \sigma_q)\}_{q \in Q}$, are derived from the maximum likelihood estimation...
 - ...by computing the mean and variance of the character pairs latency

Modeling Keystroke Pairs

The result is severe overlap of the 142 keystroke pairs



Regardless, still gain about 1.2 bits of information on average

Using the Leaked Information

- Probably can not decode an entire session to plaintext, but

Using the Leaked Information

- Probably can not decode an entire session to plaintext, but
- Concentrate on reducing the search-space of a password

Using the Leaked Information

- Probably can not decode an entire session to plaintext, but
- Concentrate on reducing the search-space of a password
- Try to guess the top n likely passwords using the inter-keystroke timing sequence

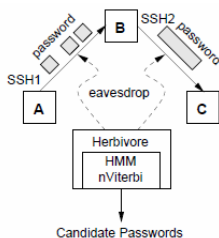
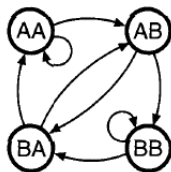


Image Credit: Dawn Song et al

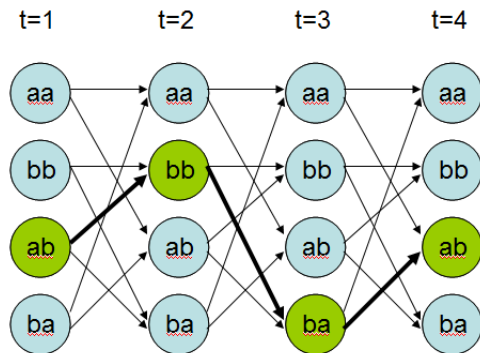
Markov Model



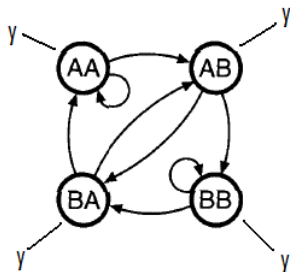
- Markov model for pressing keys a, b

Markov Model Trellis

- Example: a-b-b-a-b



Hidden Markov Model

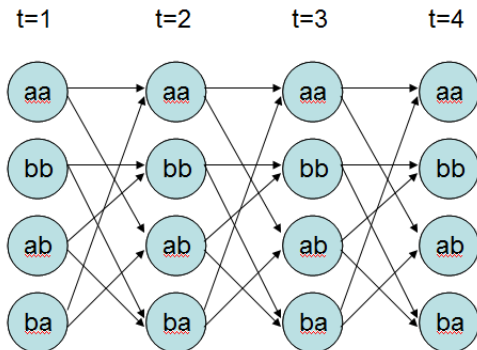


- HMM for pressing keys a, b , where we only observe key press inter-arrival times
- State transitions predefined as uniform (assume uniformly random passwords)
- Output likelihoods predefined as gaussian distributions

Image Credit: Richard Durbin et al (modified)

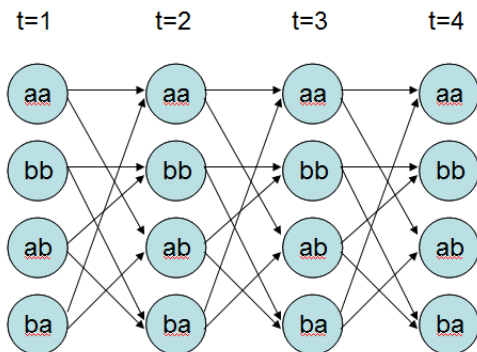
The Most Likely Sequence

- We have the same sequence: ab-bb-ba-ab
- But, only observe latency, y , for each pair



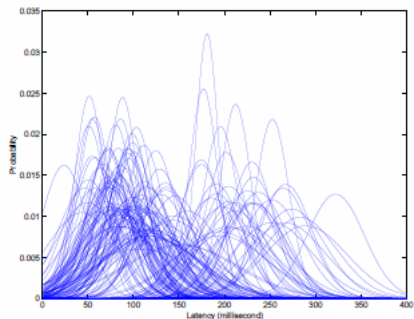
The Most Likely Sequence

- We have the same sequence: ab-bb-ba-ab
- But, only observe latency, y , for each pair



- Viterbi: $V(q_t) = \max Pr[y_t|q_t] * Pr[q_t|q_{t-1}] * V(q_{t-1})$

The Most Likely Sequence



- Unfortunately, it is highly improbable the most likely sequence will be the real sequence – need the top n sequences instead

n-Viterbi

Intuition: at each step, store the n top paths, instead of only the max

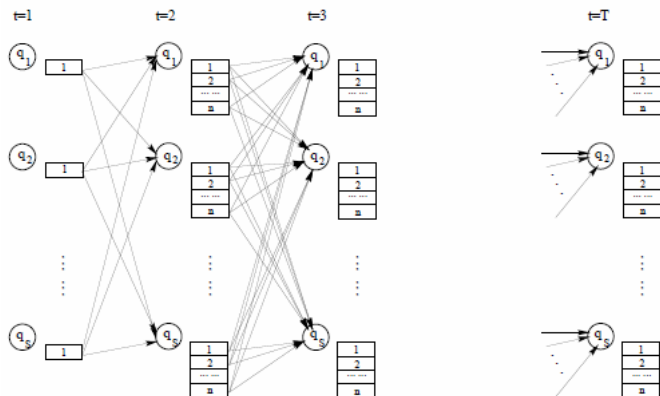


Image Credit: Dawn Song et al

Results

Training Set	Test Set	Test Cases				
		Password 1	Password 2	Password 3	Password 4	Password 5
User 1	User 1	15.6%	0.7%	2.0%	1.3%	1.6%
User 1	User 2	62.3%	15.2%	7.0%	14.8%	0.3%
User 1	User 3	6.4%	N/A	1.8%	3.1%	4.2%
User 1	User 4	1.9%	31.4%	1.1%	0.1%	28.8%
User 2	User 1	4.9%	1.3%	1.6%	12.3%	3.1%
User 2	User 2	30.8%	15.0%	2.8%	3.7%	2.9%
User 2	User 3	4.7%	N/A	5.3%	6.7%	38.4%
User 2	User 4	0.7%	16.8%	3.9%	0.6%	5.4%

- Percent of the key-space searched before finding the password

Network Protocol Classification

Paper

Charles Wright, Fabian Monrose, and Gerald Masson. *On Inferring Application Protocol Behaviors in Encrypted Network Traffic*. In Journal of Machine Learning Research, 2006

Goal

Identify the protocol of each network flow without using ports or payload inspection

Motivation

The Players

- Administrator: only wants to allow HTTP and HTTPS connections to be allowed egress
- User: wants to connect to their home box via SSH

Motivation

The Players

- Administrator: only wants to allow HTTP and HTTPS connections to be allowed egress
- User: wants to connect to their home box via SSH

Scenario 1

- Enforcement: Admin blocks any outgoing connections not on port 80 or 443
- Reaction: User runs his home SSH server on port 80

Motivation

The Players

- Administrator: only wants to allow HTTP and HTTPS connections to be allowed egress
- User: wants to connect to their home box via SSH

Scenario 1

- Enforcement: Admin blocks any outgoing connections not on port 80 or 443
- Reaction: User runs his home SSH server on port 80

Scenario 2

- Enforcement: Admin uses a payload inspector to verify protocol attributes
- Reaction: User tunnels SSH over HTTPS, which is encrypted

Demultiplexing

- Use flow-based analysis (bidirectional)

Demultiplexing

- Use flow-based analysis (bidirectional)
- Assume individual TCP connections can be demultiplexed
 - All packets that match the $(IP_{client}, IP_{server}, Port_{client}, Port_{server})$ 4-tuple are part of the same flow

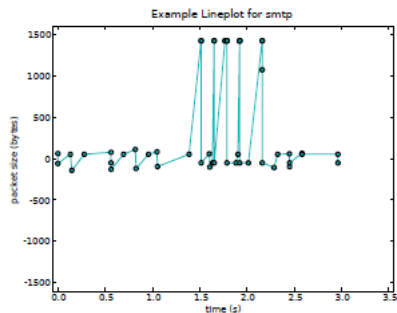
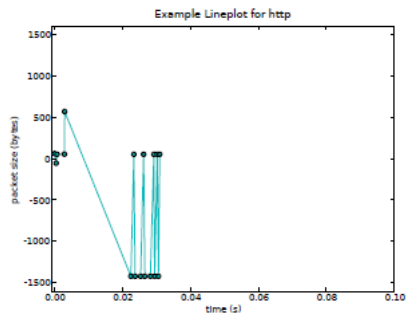
Demultiplexing

- Use flow-based analysis (bidirectional)
- Assume individual TCP connections can be demultiplexed
 - All packets that match the $(IP_{client}, IP_{server}, Port_{client}, Port_{server})$ 4-tuple are part of the same flow
 - Client vs. Server determined by 1st packet seen or TCP handshake flags

Demultiplexing

- Use flow-based analysis (bidirectional)
- Assume individual TCP connections can be demultiplexed
 - All packets that match the $(IP_{client}, IP_{server}, Port_{client}, Port_{server})$ 4-tuple are part of the same flow
 - Client vs. Server determined by 1st packet seen or TCP handshake flags
- Each packet in a flow represented by the $(client||server, inter-arrival\ time, size)$ 3-tuple

Line Plot Visualization



- Client packets: $size > 0$
- Server packets: $size < 0$

Image Credit: Charles Wright et al

An Aggregate View

- Are there distinct profiles in the general case?

An Aggregate View

- Are there distinct profiles in the general case?
- Packet distributions are not well described by a gaussian
 - too much variation in different protocols
 - multi-modal distributions

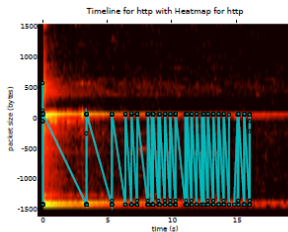
An Aggregate View

- Are there distinct profiles in the general case?
- Packet distributions are not well described by a gaussian
 - too much variation in different protocols
 - multi-modal distributions
- Need to non-parametrically infer the distribution from the data Use kernel density estimation

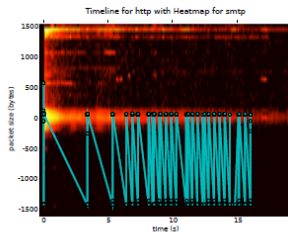
An Aggregate View

- Are there distinct profiles in the general case?
- Packet distributions are not well described by a gaussian
 - too much variation in different protocols
 - multi-modal distributions
- Need to non-parametrically infer the distribution from the data Use kernel density estimation
- The paper uses a box kernel, otherwise known as a histogram

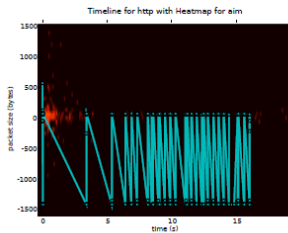
Profiling with Timeline Heatmap



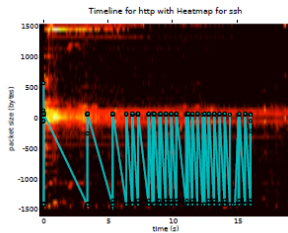
(a) HTTP



(b) SMTP



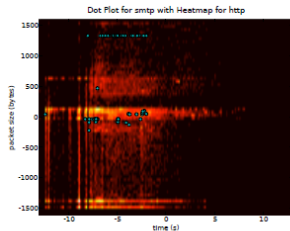
(c) AIM



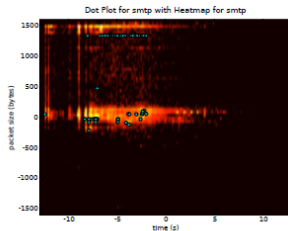
(d) SSH

Image Credit: Charles Wright et al

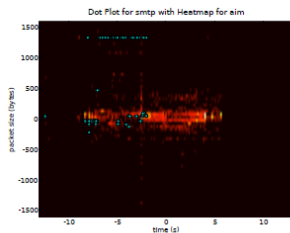
Profiling with Timeline Heatmap



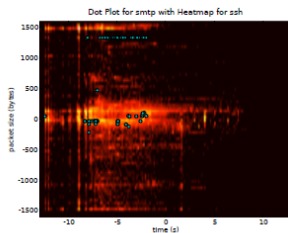
(a) HTTP



(b) SMTP



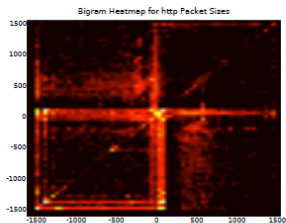
(c) AIM



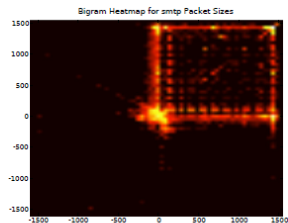
(d) SSH

Image Credit: Charles Wright et al

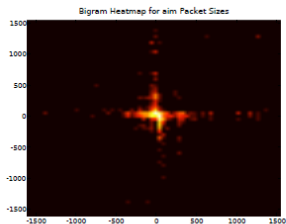
Profiling with Bigram Heatmap



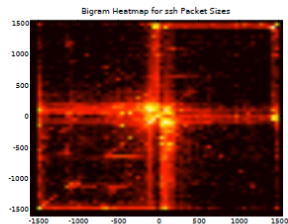
(a) HTTP



(b) SMTP



(c) AIM



(d) SSH

Image Credit: Charles Wright et al

Using the Leaked Information

- First, lets consider packet sizes only

Using the Leaked Information

- First, lets consider packet sizes only
- Model the packet sizes for packet1, packet2, etc.

Using the Leaked Information

- First, lets consider packet sizes only
- Model the packet sizes for packet1, packet2, etc.
- Now we have a sequence, use HMM!

Hidden Markov Model

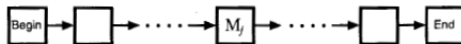


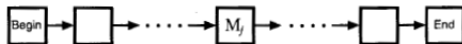
Image Credit: Richard Durbin et al

Hidden Markov Model



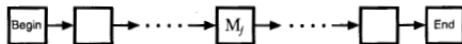
- Special issues for our problem

Hidden Markov Model



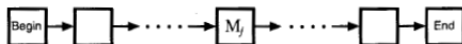
- Special issues for our problem
 - Different length flows
 - Packet retransmissions
 - Packets dropped
 - Normal variations in the protocol

Hidden Markov Model



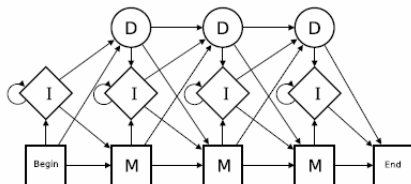
- Special issues for our problem
 - Different length flows
 - Packet retransmissions
 - Packets dropped
 - Normal variations in the protocol
- Want to profile the essential ingredients for a protocol while allowing some noise

Hidden Markov Model



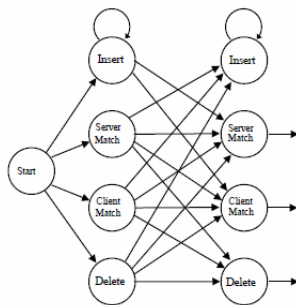
- Special issues for our problem
 - Different length flows
 - Packet retransmissions
 - Packets dropped
 - Normal variations in the protocol
- Want to profile the essential ingredients for a protocol while allowing some noise
- A standard topology to solve these problems: Profile HMMs

Standard Profile HMM



- Insert state, delete state
- A delete state transitioning to an insert state effectively represents a replacement

Profile HMM for Flows



- Direction of a packet correlated with the direction of the previous packet
- Split match state into client and server states

Generating a Profile (training)

- Initially, model parameters assigned with uniform probabilities

Generating a Profile (training)

- Initially, model parameters assigned with uniform probabilities
- Build training set by demultiplexing flows and grouping by server port

Generating a Profile (training)

- Initially, model parameters assigned with uniform probabilities
- Build training set by demultiplexing flows and grouping by server port
- Baum-Welch iteratively finds new parameters that maximize the likelihood of sequences in the training set

Model Surgery

- Some flows are 15 packets, others may be 23 packets...how to set the model length?

Model Surgery

- Some flows are 15 packets, others may be 23 packets...how to set the model length?
- The model surgery heuristic:
 - 1 Initially, length of model set to the average num of packets for a particular protocol

Model Surgery

- Some flows are 15 packets, others may be 23 packets...how to set the model length?
- The model surgery heuristic:
 - 1 Initially, length of model set to the average num of packets for a particular protocol
 - 2 Train using Baum-Welch

Model Surgery

- Some flows are 15 packets, others may be 23 packets...how to set the model length?
- The model surgery heuristic:
 - 1 Initially, length of model set to the average num of packets for a particular protocol
 - 2 Train using Baum-Welch
 - 3 Remove a state if more than half of the flows enter the delete state at a given position
 - 4 Add a state if more than half of the flows enter the insert state at a given position

Model Surgery

- Some flows are 15 packets, others may be 23 packets...how to set the model length?
- The model surgery heuristic:
 - 1 Initially, length of model set to the average num of packets for a particular protocol
 - 2 Train using Baum-Welch
 - 3 Remove a state if more than half of the flows enter the delete state at a given position
 - 4 Add a state if more than half of the flows enter the insert state at a given position
 - 5 Repeat until model stabilizes

Model Surgery

- Some flows are 15 packets, others may be 23 packets...how to set the model length?
- The model surgery heuristic:
 - 1 Initially, length of model set to the average num of packets for a particular protocol
 - 2 Train using Baum-Welch
 - 3 Remove a state if more than half of the flows enter the delete state at a given position
 - 4 Add a state if more than half of the flows enter the insert state at a given position
 - 5 Repeat until model stabilizes
- Usually only takes a few iterations

Likelihood of a Sequence (classify)

- Given set C of k classes with trained HMMs $\lambda = \lambda_1, \lambda_2, \dots, \lambda_k$, find $c \in C$ such that $c = \text{class}(O)$

Likelihood of a Sequence (classify)

- Given set C of k classes with trained HMMs $\lambda = \lambda_1, \lambda_2, \dots, \lambda_k$, find $c \in C$ such that $c = \text{class}(O)$
- Two methods:
 - 1 Find the maximum $P(O|\lambda_k)$, forward algorithm

Likelihood of a Sequence (classify)

- Given set C of k classes with trained HMMs $\lambda = \lambda_1, \lambda_2, \dots, \lambda_k$, find $c \in C$ such that $c = \text{class}(O)$
- Two methods:
 - 1 Find the maximum $P(O|\lambda_k)$, forward algorithm
 - 2 Find the maximum Viterbi path probability, $P_{\text{viterbi}}(O|\lambda_k)$, viterbi algorithm

Likelihood of a Sequence (classify)

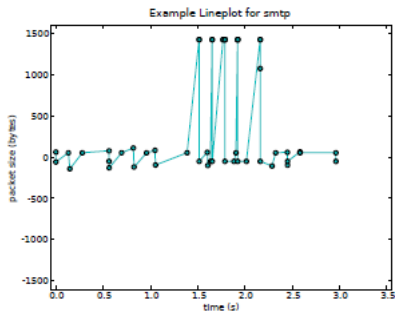
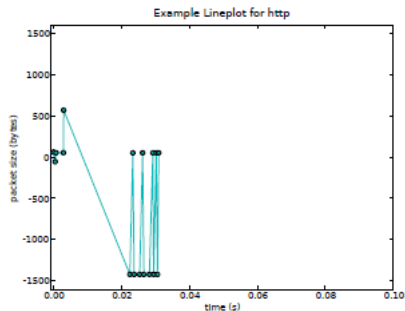
- Given set C of k classes with trained HMMs $\lambda = \lambda_1, \lambda_2, \dots, \lambda_k$, find $c \in C$ such that $c = \text{class}(O)$
- Two methods:
 - 1 Find the maximum $P(O|\lambda_k)$, forward algorithm
 - 2 Find the maximum Viterbi path probability, $P_{\text{viterbi}}(O|\lambda_k)$, viterbi algorithm
- Viterbi classifier finds each models best explanation for how the flow was generated, worked best in practice

Results Using Size

protocol	micro-level		equivalence class	
	TD	FD	TD	FD
AIM	80.80	3.41	80.80	3.41
SMTP-out	73.20	3.07	80.10	1.82
SMTP-in	77.20	4.39	87.80	3.97
HTTP	90.30	2.10	96.70	1.47
HTTPS	88.50	3.24	94.40	2.72
FTP	57.70	2.01	57.70	2.01
SSH	69.10	2.93	71.00	2.88
Telnet	82.90	3.77	86.10	4.08

Image Credit: Charles Wright et al

Incorporating Size and Timing



- In addition to size, we also have *inter-arrival time* for each packet
- Intuitively, this additional information should help
- Would like to use the same models and techniques independent of the number of features

Image Credit: Charles Wright et al

Vector Quantization

- Goal: Transform N dimensions into 1 dimension

Vector Quantization

- Goal: Transform N dimensions into 1 dimension
- Prepare data:
 - Log transform times to reduce dynamic range
 - Assign size and $\log(\text{time})$ equal weight by scaling in $[1, -1]$ range

Vector Quantization

- Goal: Transform N dimensions into 1 dimension
- Prepare data:
 - Log transform times to reduce dynamic range
 - Assign size and $\log(\text{time})$ equal weight by scaling in $[1, -1]$ range
- Split into client and server vectors

Vector Quantization

- Goal: Transform N dimensions into 1 dimension
- Prepare data:
 - Log transform times to reduce dynamic range
 - Assign size and $\log(\text{time})$ equal weight by scaling in $[1, -1]$ range
- Split into client and server vectors
- Reduce dimensions:
 - 1 Randomly select k vectors as cluster centroids

Vector Quantization

- Goal: Transform N dimensions into 1 dimension
- Prepare data:
 - Log transform times to reduce dynamic range
 - Assign size and $\log(\text{time})$ equal weight by scaling in $[1, -1]$ range
- Split into client and server vectors
- Reduce dimensions:
 - 1 Randomly select k vectors as cluster centroids
 - 2 Assign each vector to the nearest centroid

Vector Quantization

- Goal: Transform N dimensions into 1 dimension
- Prepare data:
 - Log transform times to reduce dynamic range
 - Assign size and $\log(\text{time})$ equal weight by scaling in $[1, -1]$ range
- Split into client and server vectors
- Reduce dimensions:
 - 1 Randomly select k vectors as cluster centroids
 - 2 Assign each vector to the nearest centroid
 - 3 Recalculate centroids

Vector Quantization

- Goal: Transform N dimensions into 1 dimension
- Prepare data:
 - Log transform times to reduce dynamic range
 - Assign size and $\log(\text{time})$ equal weight by scaling in $[1, -1]$ range
- Split into client and server vectors
- Reduce dimensions:
 - 1 Randomly select k vectors as cluster centroids
 - 2 Assign each vector to the nearest centroid
 - 3 Recalculate centroids
 - 4 Repeat until reassignment drops below some threshold

Vector Quantization

- Goal: Transform N dimensions into 1 dimension
- Prepare data:
 - Log transform times to reduce dynamic range
 - Assign size and $\log(\text{time})$ equal weight by scaling in $[1, -1]$ range
- Split into client and server vectors
- Reduce dimensions:
 - 1 Randomly select k vectors as cluster centroids
 - 2 Assign each vector to the nearest centroid
 - 3 Recalculate centroids
 - 4 Repeat until reassignment drops below some threshold
- Now each N dimensional vector represented as one of k codewords

Vector Quantization

- Goal: Transform N dimensions into 1 dimension
- Prepare data:
 - Log transform times to reduce dynamic range
 - Assign size and $\log(\text{time})$ equal weight by scaling in $[1, -1]$ range
- Split into client and server vectors
- Reduce dimensions:
 - 1 Randomly select k vectors as cluster centroids
 - 2 Assign each vector to the nearest centroid
 - 3 Recalculate centroids
 - 4 Repeat until reassignment drops below some threshold
- Now each N dimensional vector represented as one of k codewords
- New vectors are assigned the nearest codeword

Vector Quantization

- Goal: Transform N dimensions into 1 dimension
- Prepare data:
 - Log transform times to reduce dynamic range
 - Assign size and $\log(\text{time})$ equal weight by scaling in $[1, -1]$ range
- Split into client and server vectors
- Reduce dimensions:
 - 1 Randomly select k vectors as cluster centroids
 - 2 Assign each vector to the nearest centroid
 - 3 Recalculate centroids
 - 4 Repeat until reassignment drops below some threshold
- Now each N dimensional vector represented as one of k codewords
- New vectors are assigned the nearest codeword
- Finally, follow the same process as before using codewords instead of sizes as the HMM output symbols

Results Using Size

protocol	micro-level		equivalence class	
	TD	FD	TD	FD
AIM	80.80	3.41	80.80	3.41
SMTP-out	73.20	3.07	80.10	1.82
SMTP-in	77.20	4.39	87.80	3.97
HTTP	90.30	2.10	96.70	1.47
HTTPS	88.50	3.24	94.40	2.72
FTP	57.70	2.01	57.70	2.01
SSH	69.10	2.93	71.00	2.88
Telnet	82.90	3.77	86.10	4.08

protocol	micro-level		equivalence class	
	TD	FD	TD	FD
AIM	83.90	2.53	83.90	2.53
SMTP-out	74.40	2.24	79.70	1.60
SMTP-in	79.80	3.34	85.90	3.02
HTTP	78.00	1.09	92.90	0.62
HTTPS	87.20	3.74	91.10	1.88
FTP	58.20	1.81	58.20	1.81
SSH	76.30	8.37	77.80	7.90
Telnet	79.50	2.44	90.70	2.60

VoIP Phrase Recognition

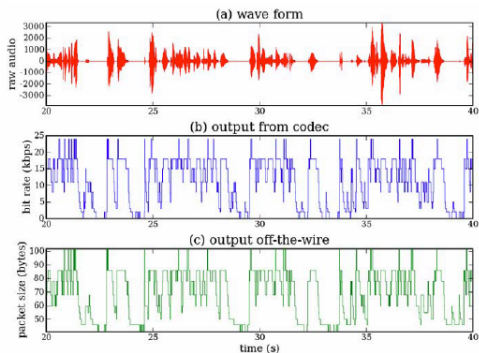
Paper

Charles Wright, Lucas Ballard, Scott Coulls, Fabian Monrose, and Gerald Masson. *Spot me if you can: recovering spoken phrases in encrypted VoIP conversations*. IEEE Security and Privacy, 2008

Goal

Recognize when certain phrases are spoken on an encrypted VoIP phone call

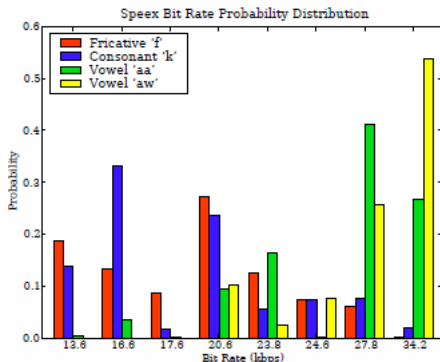
Waveform to Packet



- Variable bit rate (VBR) encoding uses a variable number of bits in its codebook, depending on the sound
- Property is propagated when stream cipher is used for encryption

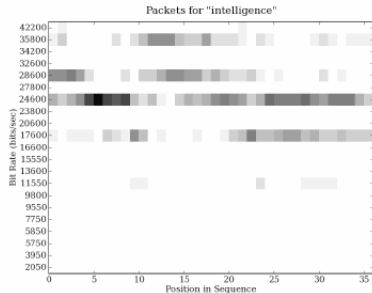
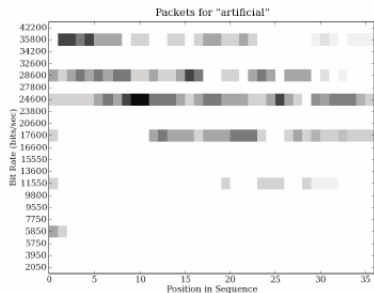
Image Credit: Charles Wright et al

Bit Rate vs Phoneme



- By VBR encoding utterances of several phonemes, we see they are correlated with the bit rate of the encoding

Bit Rate vs Phoneme



- Several utterances of artificial and intelligence encoded and packetized
- Dark areas indicate the correlation of sound to packet size

Image Credit: Charles Wright et al

Using the Leaked Information

- Very similar to flow classification

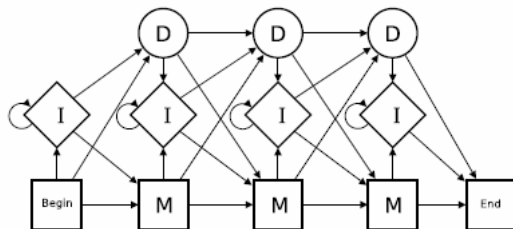
Using the Leaked Information

- Very similar to flow classification
- Build profiles for each phrase instead of each protocol

Using the Leaked Information

- Very similar to flow classification
- Build profiles for each phrase instead of each protocol
- We model the packet sizes for packet1, packet2, etc.

Profile HMM



- Like with flows, spoken phrases tend to vary
 - Different speakers
 - Variability with same speaker
 - Added variance due to adaptive compression
- Again, a profile HMM is suited to handle these issues

Estimating Model Parameters (training)

- First, need a training set with many instances of the target phrase

Estimating Model Parameters (training)

- First, need a training set with many instances of the target phrase
- Any word can be built with a combination of phonemes using a pronunciation dictionary

Estimating Model Parameters (training)

- First, need a training set with many instances of the target phrase
- Any word can be built with a combination of phonemes using a pronunciation dictionary
- Use dictionary in conjunction with a database of phoneme utterances

Estimating Model Parameters (training)

- First, need a training set with many instances of the target phrase
- Any word can be built with a combination of phonemes using a pronunciation dictionary
- Use dictionary in conjunction with a database of phoneme utterances
- Artificially build many variations of a phrase, encode, and packetize
Now train...

Estimating Model Parameters (training)

- Match state emission probabilities set uniformly random

Estimating Model Parameters (training)

- Match state emission probabilities set uniformly random
- Probability of transitioning to a Match state set much high than to any other state

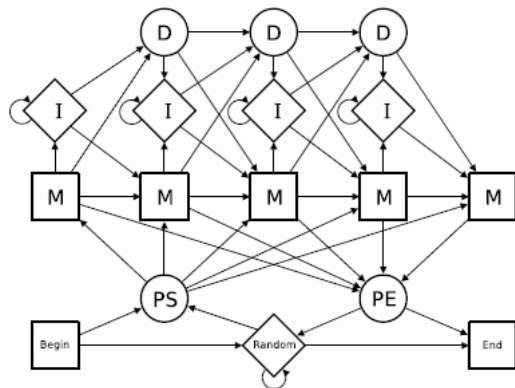
Estimating Model Parameters (training)

- Match state emission probabilities set uniformly random
- Probability of transitioning to a Match state set much high than to any other state
- Apply the Baum-Welch algorithm using simulated annealing, attempting to reduce local maxima

Estimating Model Parameters (training)

- Match state emission probabilities set uniformly random
- Probability of transitioning to a Match state set much high than to any other state
- Apply the Baum-Welch algorithm using simulated annealing, attempting to reduce local maxima
- Finally, use viterbi training

Finding ... In a Stream of Packets



- Phrases will be surrounded by packets representing the rest of the conversation
- Random state matches packets not part of the phrase

Classification

- Use Viterbi algorithm to find most likely state sequence

Classification

- Use Viterbi algorithm to find most likely state sequence
- Consider each subsequence in the profile part of the model a hit

Classification

- Use Viterbi algorithm to find most likely state sequence
- Consider each subsequence in the profile part of the model a hit
- Calculate the probability of the hit being in the profile vs random state
 - $score_{hit} = \frac{P(hit|Profile)}{P(hit|Random)}$
- Accept a hit with a score above some threshold

Results

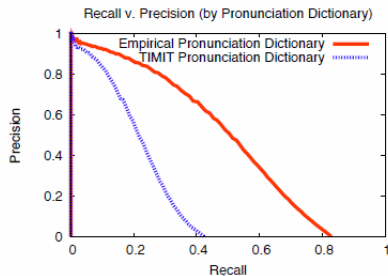
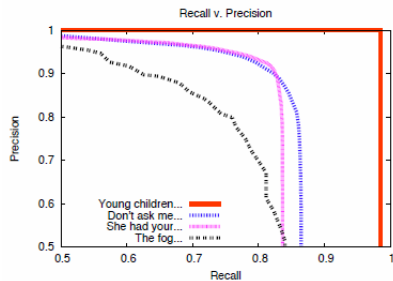


Image Credit: Charles Wright et al

Results

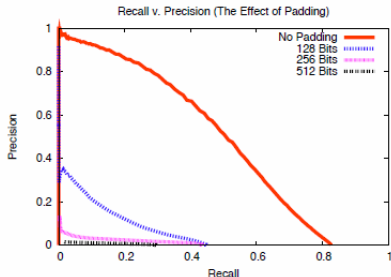
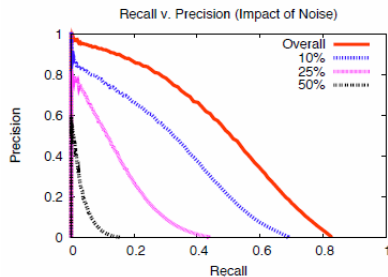


Image Credit: Charles Wright et al

End