Advances in Internet of Things, 2011, 1, 13-26
doi:10.4236/ait.2011.12003 Published Online July 2011 (http://www.SciRP.org/journal/ait)

Side-Channel Analysis for Detecting Protocol Tunneling

Harakrishnan Bhanu[1], Jason Schwier[1], Ryan Craven[1], Richard R. Brooks[1], Kathryn Hempstalk[2], Daniele Gunetti[3], Christopher Griffin[4]
[1]*Holcombe Department of Electrical and Computer Engineering*, *Clemson University*, *Clemson*, *USA*
[2]*Department of Computer*, *The University of Waikato*, *Hamilton*, *USA*
[3]*Dipartimento di Informatica*, *Università degli Studi di Torin*, *Corso Svizzera*, *Torino*, *Italy*
[4]*Department of Mathematics*, *Pennsylvania State University*, *University Park*, *USA*
E-mail: *rrb@clemson.edu*
Received May 28, 2011; *revised June* 30, 2011; *accepted July* 8, 2011

Abstract

Protocol tunneling is widely used to add security and/or privacy to Internet applications. Recent research has exposed side channel vulnerabilities that leak information about tunneled protocols. We first discuss the timing side channels that have been found in protocol tunneling tools. We then show how to infer Hidden Markov models (HMMs) of network protocols from timing data and use the HMMs to detect when protocols are active. Unlike previous work, the HMM approach we present requires no a priori knowledge of the protocol. To illustrate the utility of this approach, we detect the use of English or Italian in interactive SSH sessions. For this example application, keystroke-timing data associates inter-packet delays with keystrokes. We first use clustering to extract discrete information from continuous timing data. We use discrete symbols to infer a HMM model, and finally use statistical tests to determine if the observed timing is consistent with the language typing statistics. In our tests, if the correct window size is used, fewer than 2% of data windows are incorrectly identified. Experimental verification shows that on-line detection of language use in interactive encrypted protocol tunnels is reliable. We compare maximum likelihood and statistical hypothesis testing for detecting protocol tunneling. We also discuss how this approach is useful in monitoring mix networks like The Onion Router (Tor).

Keywords: Hidden Markov Models, Timing Side-Channel Attack, VPN Vulnerability

1. Introduction

Communications protocols are typically described using either the 7-layer Open Systems Interconnect (OSI) model from the International Standards Organization (ISO) or the four layer Internet Engineering Task Force (IETF) Internet Protocol (IP) stack [1]. In both, each layer of the network stack, except the lowest physical layer, is a set of network protocols recursively tunneled within protocols at lower layers of the stack. This creates an adaptable design space where functionality missing at one layer can be provided at another layer of the stack.

The original IP design largely ignored security issues. IPv4 sends data in clear text and data packets are not authenticated. Security is typically added to IP by either:

- Using virtual private networks (VPNs) that add security by tunneling standard IP packets through an encrypted virtual network connection [2], or
- Using IPsec, which is part of the IPv6 standards. A major protocol in the IPSec suite is essentially a VPN that tunnels IPv4 packets through encrypted connections [2].

Mix networks like The Onion Router (Tor) and Invisible Internet Protocol (I2P) use tunneling to add both anonymity and security to IP [3].

Protocol tunneling can also be used to evade network security enforcement. Protocol tunneling through SSH[1], HTTP, and even DNS has been used to circumvent security enforcement by firewalls and application layer gateways [4]. Since packets tunneled through SSH or SSL are encrypted, security enforcement tools are effectively unable to inspect their contents [4]. One impediment to

[1]SSH is a standard Internet tool that opens a shell on a remote machine and secures interactions by encrypting the data stream. In this paper, tests used the current version of SSH; SSH 2.0. Previous work [1] used **SSH** v1.0.

IPv6 adoption is the fact that mandatory use of encrypted IPsec tunnels disables firewall deep packet inspection.

On the other hand, current protocol tunneling tools have been found vulnerable to side-channel attacks, which do not directly compromise security measures. Instead, side-channels extract information by indirectly observing implementation artifacts. For example, a significant timing side-channel vulnerability for SSH can extract the system password from interactive sessions [5]. This paper discusses how side-channel analysis can detect the presence of protocols within tunneled connections.

There are three main contributions of this paper. First, it provides a brief, up-to-date survey of current side-channel attacks on tunneled protocols. Second, an application is presented that provides a practical example of how tunneled protocols can be detected. Third, the approach we present has significant advantages over previous approaches. It makes minimal assumptions about the protocol being attacked and requires no *a priori* information about the protocol. The use of statistical hypothesis testing, instead of maximum likelihood comparisons, has multiple advantages that are explained in Section 3.

The rest of this paper is organized as follows. Section 2 provides a brief survey of current research literature on side-channel vulnerabilities in tunneled protocols that use security tools like SSH, SSL, Tor, and I2P. Section 3 describes our approach to protocol detection. It uses zero-knowledge hidden Markov model inference [6,7] to extract models of network protocols from observed timing data. To illustrate how protocol detection works, we present an example application in Section 4. We show how inferred HMMs detect the language used in interactive remote sessions tunneled through SSH. Section V analyzes the performance of our example application. Section VI discusses our results and presents our conclusions.

2. Brief Survey of Side-Channel Attacks on Tunneled Protocols

Figure 1 illustrates how protocol tunneling is done by virtual private networks. For example, many Linux VPNs create a virtual network interface TUN [2]. Applications connect to TUN, like they would connect to the Ethernet network interface. TUN multiplexes these sessions, encrypts them, and passes them to the real network interface. The network interface adds a routing header to the encrypted TUN packet and sends this new packet over the Internet. The network interface on the destination node removes the routing information and forwards the packet payload to its local TUN interface, which demultiplexes and decrypts the sessions. This adds security to IP networks by encrypting the communications sessions and providing a private name space.

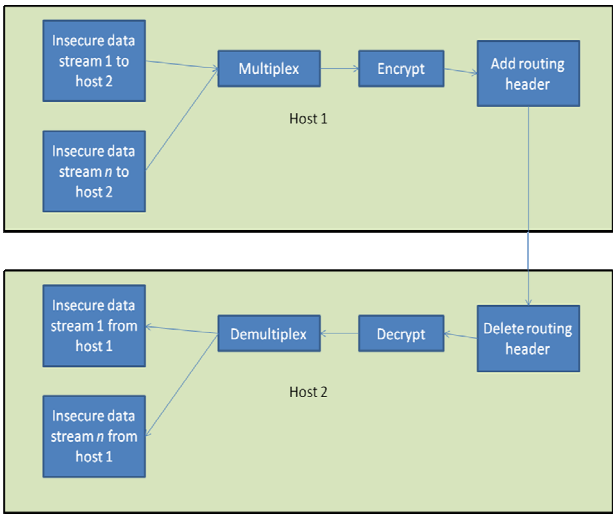

Figure 1. Notional illustration of protocol tunneling.

For this procedure to work on the current Internet, routing information is not encrypted, which makes the system vulnerable to traffic analysis. **Figure 2** shows how The Onion Router (Tor) attempts to make communications immune to traffic monitoring in order to provide anonymous communications [8,9]. The connection source (Alice) contacts a Tor directory node for a list of cooperating Tor relays. Alice chooses (typically) three Tor relay nodes. The final Tor node acts as a proxy for Alice in her connection with destination (Bob). Each hop within Tor is a tunneled connection encrypted with a different key. Each intermediate Tor node is only aware of its session key, the previous node, and the next node in the session relay. For each packet it receives, it strips off the routing data, decrypts the encapsulated packet, and relays the results of the decryption to the next node (see **Figure 1**).

I2P uses an approach like Tor with a few additions [10,11]. User communications go through an encrypted proxy address embedded in the I2P cloud. Each user can

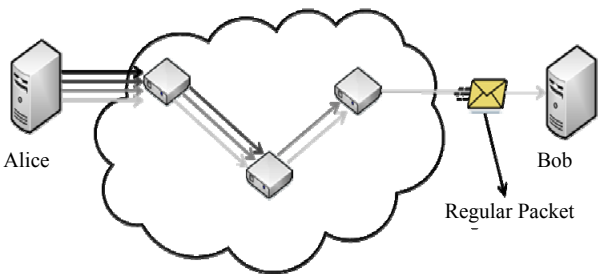

Figure 2. Anonymous tunneling through Tor or I2P.

have multiple active sessions multiplexed through the cloud. Where Tor has only a low latency mode that forwards packets as they arrive, I2P also provides medium and high latency modes.

In addition to tunneling using encryption tools like SSH or SSL, sessions can be tunneled through other protocols like http and DNS [4]. This can go to extremes. DNS has been tunneled through SSH and SSH through DNS, leading to the recursive tunneling of DNS through DNS [12].

Protocol tunneling makes network monitoring difficult. When the tunnels are protected using encryption, it becomes necessary to either break the encryption scheme using cryptanalysis or resort to side-channel attacks. Since modern encryption protocols are designed to be prohibitively expensive to attack[2], side-channel attacks that exploit implementation artifacts are attractive.

For example, web browsers use SSL to encrypt network traffic and protect user information from exposure. Unfortunately modern web pages consist of many components, such as CSS style sheets, images, etc. The side-channel attack in [14] uses this information to determine the sequence of web pages viewed by a user, even when SSL encrypts the network traffic. While encryption effectively secures information entered into forms on web pages, the packet sizes of the SSL encrypted packets can be monitored. These sizes correspond directly to web page elements and provide enough information to typically track the sequence of web pages visited during a user's web browsing session.

To maintain Quality of Service, SSH transmits keystrokes as they are typed, preserving the inter-keystroke delays. One packet is transmitted for each keystroke from the user's local machine to the remote host. Song *et al*. [5] use timing analysis to infer the system password from interactive SSH sessions. They manually constructed an HMM to represent interactive SSH sessions and trained the HMM using data collected by observing the individual doing remote system administration tasks. They then used a priori information particular to SSH version 1 to determine exactly which keystrokes correspond to the password. The timing data corresponding to the sequence of characters for the password can then be entered into the trained HMM. An n-Viterbi algorithm is then used to find the n character sequences that most likely produced the timing signature. This information reduces the computational effort required to infer the password using traditional techniques by a factor of 50. This basic vulnerability is present in most secure communications applications, including virtual private networks built on SSH cryptography protecting tunneled connections.

A similar approach has been used to identify specific phrases in encrypted voice over IP (VOIP) traffic [15]. When variable bit rate encoding is used by the VOIP codec, the encodings of different classes of phonemes require predictable ranges of bit sizes; producing a vulnerability similar to the one in [14]. As in [5] in this attack, HMMs are constructed and trained to recognize specific phases of interest. They determine the Viterbi path of observed phrases through the HMM and use a log-likelihood ratio metric comparing the observed path with random noise. Their test shows that phrases can be identified in encrypted data streams with probabilities ranging from 50% to 90%.

The Tunnel-Hunter approach [4] uses both inter-packet arrival rates and packet sizes to define protocol profiles. Training data is collected for one or more protocols tunneled through either SSH or http. As a network is monitored, observation data is used to construct a profile matrix. Instead of using HMMs for protocol detection, they use Bayesian techniques to create maximum likelihood classifiers. If a timing profile matches a class of forbidden applications, then the tunneled session is terminated. The test results for this approach are promising.

We now describe side-channel attacks on traffic using mix networks. Several researchers have used timing information to attack the anonymity provided by mix networks like Tor and I2P. If attackers can collect inter-packet timing information at all network entry and exit points, interpacket timings can be cross-correlated to calculate the mutual information between entry-exit pairs. This reliably identifies the correct communications paths using sample sizes on the order of seconds to tens of seconds. Surprisingly, this attack works better on larger networks than on smaller ones [16]. This is an instance of the more general class of flow correlation attacks [17]. This type of attack does not have to be constrained to entry/exit nodes. The work in [18] analyzes traffic at global choke points to determine the global region where a given service is hosted. These attacks can best be countered by saturating the communications channels [19] leaving no available bandwidth for patterns to emerge. Due to the extreme resource requirements of channel saturation, this approach can only be used in extreme cases.

The side-channel attacks presented thus far were all passive. It is also possible to extract timing information by either using malicious Tor nodes or actively inserting traffic into the network. One early approach inserted a malicious node into Tor [20], made the node attractive

[2]Current implementations of tools like **SSL** do have vulnerabilities, but these are mainly due to deficiencies in implementation or supporting infrastructure [13], they are rarely due to deficiencies in the cryptographic algorithm.

for use as a contact node, and used packet counting to identify the real identity of a node trying to be anonymous. In another approach a malicious node in the Tor network inserts traffic flow that deliberately slows down intermediate nodes [21]. By correlating the traffic flow of the session of interest with the disturbance traffic it becomes possible to identify the nodes that are being used as intermediates in the Tor session. This attack is called low-cost, since it does not require monitoring the entire network, like [16,17]. The congestion attack in [21] worked well in the small prototype Tor network, but fails to scale. To overcome these failings, it has been extended [21] in two ways. The first extension created long circular paths within Tor to generate congestion traffic. The second extension assumed that the user selects a malicious proxy exit node. The proxy could then modify webpages being retrieved to include malicious Javascript code that generates traffic to help trace node traffic. A similar idea is presented in [22] which localizes client nodes by measuring the time difference between when a specific web-page is returned and the client requests an object embedded within the web-page. This attack allows nodes within Tor to determine if two sessions to the same host started at the same client.

Suggestions for countering these active attacks [13,21] include removing the ability of participating nodes to discover the full list of participating nodes, introducing higher latency communications modes, and adding garbage traffic to obscure patterns [23,24]. These suggestions are problematic. If participating nodes are not addressable by malicious insiders, they will also be unavailable for use by legitimate users. I2P does interleave higher latency traffic with low-latency traffic, which is a possible countermeasure but Tor provides low-latency connections for usability reasons. Finally, adding random noise typically does not counteract correlation attacks; at best it increases the sample size necessary to reliably identify communications patterns.

Timing side-channels are not always due to network latency. The timing skew due to changes in processor clock speeds can be remotely detected [25,26]. In [25] a machine is expected of hosting a service anonymized by Tor. Large volumes of traffic are requested by one node while another node continuously pings the suspect node. The heat generated by the additional workload detectably changes the processor's clock speed, which is easily detected in the ping messages. This basic attack is extended in [26], which no longer requires inducing a large workload. It is possible to simply detect the patterns in the clock variation, which produce a detectable fingerprint. This fingerprint can also be used to geo-locate the hidden service by correlating clock skew with the time of day and temperature variations.

3. HMM Inference

The approach we use resembles [5,16] in that we use HMM models to analyze side-channel information. However, we extend their work in important ways. As long as the protocol to be detected can be expressed with a finite number of states and state transition probabilities are stationary, our approach is valid [27]. Instead of using standard HMM approaches that require an a priori known state space for training, we require no prior knowledge. Instead of using maximum likelihood metrics [4,5,15-17], we combine HMMs with statistical hypothesis testing which provides a theoretical basis for determining threshold values [28-30]. Hypothesis testing allows us to determine the statistical significance of the inferred model, which in turn indicates whether or not the volume of training data is sufficient [28-30]. Maximum likelihood approaches also typically consider all the available observed data for making a decision, which has a number of drawbacks. If the number of observations is large, the likelihood value computed by maximum likelihood is subject to underflow. The underflow danger can be countered by frequently renormalizing the likelihood value, which decreases the precision of the value being computed. In contrast, the values we use [28] become more precise as the volume of observation data increases. Our approach calculates values over a sliding window of observations. Methods for calculating window size are given in [29,30].

In the rest of this section, we describe our HMM inference procedure. Section 3.1 we describe how to extract classes of observations from continuous timing data. We show how to extract HMMs from training data in Section 3.2. Section 3.3 explains the stopping criteria for this process, which results in either producing a significant model or collecting more training data. We conclude this discussion with Section 3.4 explaining how the models are used for protocol detection. Section 4 will explain how this approach detects the languages used in interactive SSH sessions. Consider the language typed as an example of a complex network protocol.

3.1. Observation Class Inference

As Song *et al*. demonstrated, the delays between keystrokes are preserved when using an SSH tunnel [5]. This can be exploited without explicitly attacking the cryptographic protocol by analyzing the sequence of delays in the SSH data stream to detect behaviors. In our example application, we extract typing statistics from data sets of typing behaviors collected from native speakers of English [31] and Italian [32]. We then use knowledge of the delays between specific keystroke pairs (ex. "a then s" vs.

"a then p") to associate inter-packet delays in the SSH stream to a set of potential clear-text equivalents.

We use the HMM inference approach discussed in the remainder of this section to extract a Markov model of the conditional probabilities inherent in English and Italian. For example, in English once the letter q has been typed it is much more likely to be followed by the letter u than the letter z. We used two methods to associate observed packet delays with pairs of symbols.

HMM processing is based on symbolic observations. For timing analysis, we need to find the classes of observations that best represent the data we collected. We first collected statistically significant data sets of native speakers of Italian typing Italian texts [32] and native speakers of English typing English texts [31]. We verified that the volume of data we had was sufficient and that the variance between typists was not large enough to invalidate our results.

We initially plotted the Normal distribution models of key-pair timing data using the means and variances extracted from the data. These plots are shown in **Figures 3** and **4**. Since the overlaps are too large to effectively distinguish between key-pairs, a clustering approach was used [33] to find distinct classes of key-pairs. Growing Neural Gas (GNG) identified 4 clusters for the Italian keystroke data and 10 for the English. Ranges were determined as shown in **Table 1**. In this paper, GNG happens to be the clustering algorithm that was used. We expect that other approaches, such as k-means clustering, self-organizing maps, etc. would probably have produced similar results. A more rigorous explanation of this symbolization process is in [33].

3.2. Hidden Markov Model Inference

States of a Hidden Markov Models (HMMs) are not directly observed. Instead, state outputs are observed. Unlike common HMMs inference approaches [34], our approach directly associates state output symbols with state transitions.

In [7] Schwier *et al.* show it is possible to construct HMMs without a priori knowledge of the system [27] to find patterns in a symbolic time series. We use the conditional probabilities in data streams to infer a state space [27]. The algorithm starts by dividing the training data set into segments of length two. It computes the conditional probabilities present in the data—e.g., P(u|q)—resulting in a conditional probability density function for each symbol observed. A χ^2-square test[3] at the desired confidence level finds a set of unique probability distribution functions (pdfs). This set is the initial state space. In this work, we use a confidence level of 0.95.

We then consider training data segments of increasing string length L. For example, with a string length L = 3 and a two symbol (A and B) alphabet, the algorithm would compute conditional probabilities for BB, AB, BA, and AA, being followed by an A or a B. Each unique pdf is a state and the set of sequences of symbols that lead to that state is its history [27]. The values of the pdf associated with the state become the transition probabilities leading to new states. This process continues until a pre-

Table 1. Symbolization of English and Italian keystroke statistics. The means shown in the table above are those identified by Growing Neural Gas (GNG). Bounds were determined by locating the midpoint between means. These bounds were then associated with symbols. All values in the table are in milliseconds. An upper bound of 10 seconds was used to prevent any symbols from being identified as "null".

English				Italian			
L. Bound	U. Bound	Mean	Symbol	L. Bound	U. Bound	Mean	Symbol
0.00	125.00	95.14	A	15.32	0.00	28.00	A
126.00	182.00	153.17	B	38.88	29.00	45.00	B
183.00	236.00	209.04	C	49.98	46.00	59.00	C
237.00	287.00	261.29	D	67.19	60.00	10000.00	D
288.00	329.00	311.21	E				
330.00	364.00	345.07	F				
365.00	414.00	382.01	G				
415.00	494.00	445.05	H				
495.00	625.00	541.29	I				
626.00	10000.00	707.73	J				

[3]Each conditional probability is a Conditional random function. If the training data is sufficiently large, each Conditional pdf converges to a Multi-Variate Normal pdf by the Central Limit Theorem. The chi-square test is the standard test for determining if two sets of conditional probabilities are not the same.

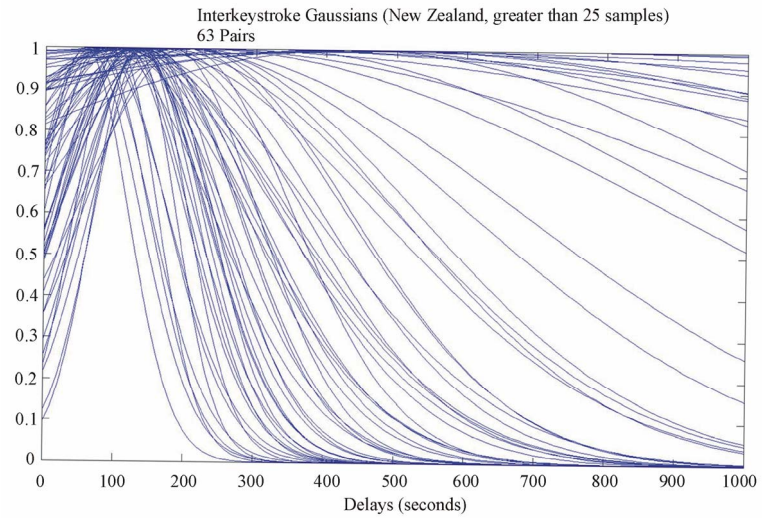

Figure 3. English Keystroke Gaussian.

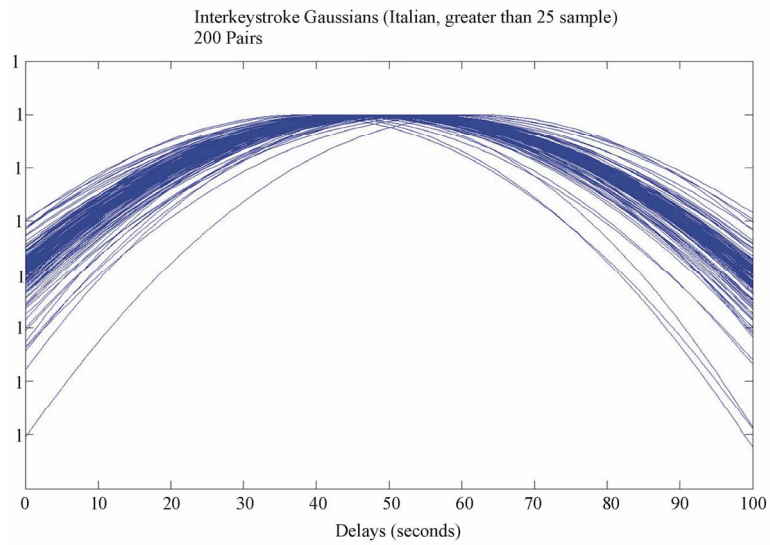

Figure 4. Italian Keystroke Gaussian.

scribed value of L is reached. We refer the reader to [6,7,27] for details about the CSSR algorithm and [6,28] to find L for a given process.

3.3. Stopping Criteria

As discussed in [6,28], HMM inference continues until one of several conditions occurs; first, we infer the HMM. We then test the inferred HMM for statistical significance using the process in [6]. This considers each conditional probability as a collection of Bernoulli random functions. We verify that the sample sets used for determining the values of these probabilities were sufficiently large. If not, we determine how much additional data is required and restart the process with a larger training set. If sample sizes are sufficient, CSSR is repeated with the string length incremented by one. If this model is also statistically significant, and identical to the model produced by the earlier iteration, the process has converged. As explained in [7], the HMM changes as the length of the training string grows until we reach the point where the correct model has been found. **Figure 5** gives a flowchart of this process.

3.4. Problem Detection

Unlike traditional HMMs [34], the models we infer have no starting state. To determine whether a HMM is consistent with an observed symbol sequence, all starting states are considered. If a symbol occurs in a state without a corresponding transition, the combination of HMM and start state is rejected.

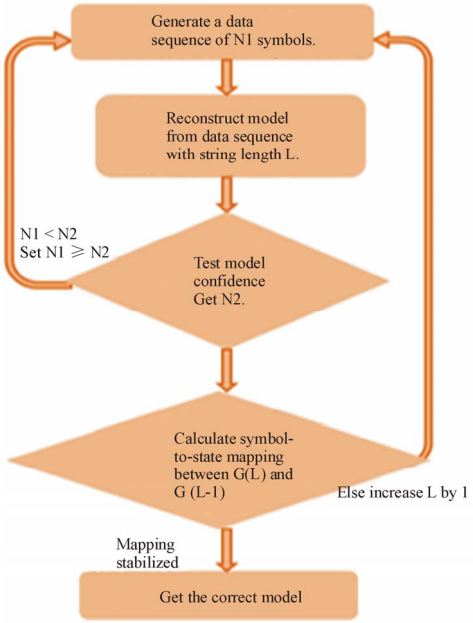

Figure 5. How to determine HMM inference stopping criteria.

To determine if a model produced a symbolized sequence, the maximum-likelihood forward-backward method is typically used [34]. The forward-backward procedure solves a classification problem; we are concerned with detection. We discuss classification versus detection in Section IV.

To solve the detection problem, we use the confidence interval approach from [28]. The confidence interval approach counts the number of times a particular state is entered and creates confidence interval bounds for each exiting transition by dividing exiting transition counts by the entering transition count. The confidence interval for the transition in question can then be found from Expression (1).

$$\left[p_{i,j} - Z_{\alpha/2}\sqrt{\frac{p_{i,j}(1-p_{i,j})}{c_i}}, p_{i,j} + Z_{\alpha/2}\sqrt{\frac{p_{i,j}(1-p_{i,j})}{c_i}} \right] \quad (1)$$

where $p_{i,j}$ is the transition probability from state i to state j for a fixed symbol, c_i is the entry-counter for state i, and $Z_{\alpha/2}$ is taken from the standard Normal distribution. These probabilities were known to us, as we constructed the models.

If the estimate falls within the confidence interval, we accept it as being correct with a false positive rate of α. If the frequencies, and hence the transition probabilities, do not fall within this range, the model is rejected as it should not have generated the string.

To map transition acceptance or failure across the model, we follow the approach from [28] and use Receiver Operating Characteristic (ROC) curves to determine the ideal threshold for acceptance of false positives. This is done by identifying the point on the curve nearest to the point (0,1), corresponding to 0% false positives, 100% true positives. By allowing a false positive rate equal to the threshold value, the true positive rate is maximized. Consequently, if the rejection rate exceeds this threshold, the model is similarly rejected. This is because more false positives were encountered than ideal. However, if the acceptance rate passes this threshold, the model is accepted as a valid source for the presented symbol sequence.

4. Language Detection

Our language structure HMMs were inferred from keystroke data [31,32] collected from native speakers of English and Italian using their native keyboards. We extracted the keystroke dynamics of each language. However, the data did not include statistically significant samples of all key-pairs. One data set did not include upper case data. The keystroke pairs for which sufficient data was available were classified by source and destination key. Their means and variances were determined.

For key-pairs where sufficient data were not available, interpolation was performed: if the key-pair AU had no samples, the delays for surrounding key-pairs AY, AJ, AI and A7 were averaged. If none of these were present, then QU, SU and ZU were averaged. That is to say, first the neighbors of the destination key (U) were considered, and then those of the source key (A). For reference, keyboard layouts for New Zealand and Italy are shown in **Figures 6** and **7**.

Training data for HMM construction were collected from Project Gutenberg. Recent, (1900 or later), texts were taken and preprocessed to remove case and special characters. Training and testing data sets were established. The zero-knowledge approach from [7] was used to extract HMMs from the training set. The resulting HMMs are shown in **Figures 8** and **9**.

For the Italian data, a reconstruction with a string length L = 3 was possible. We could only use L = 1 for the English data. Our training sets had approximately 1.1 million key-pairs. Our clustering approach gave us 10 distinct key-pair clusters for English. We used the approach from [6] to determine both the significance of the models and the volume of data necessary for creating a significant model. Creating a significant model for L = 1 would have required a training set of over 11 million key-pairs. This was due to the existence of a number of low probability transition events. We were forced to stop HMM inference and use the approximate model that we inferred with 1.1 million key-pairs and L = 1. Therefore, the English model only considers conditional probability

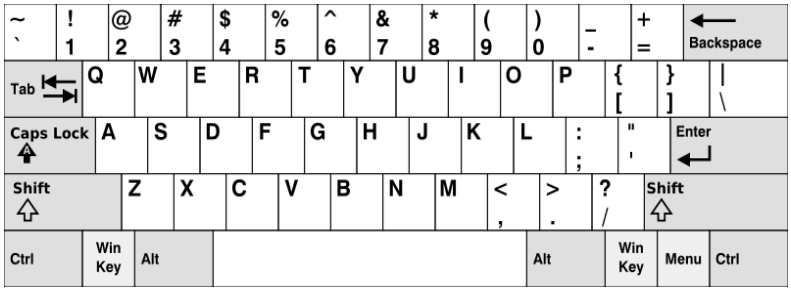

Figure 6. New Zealand Keyboard Layout (Source: http://wapedia.mobi/en/File:KB_United_States-NoAltGr.svg). Reproduced from Wapedia under the Creative Commons Attribution/Share-Alike License and GNU Free Documentation License.

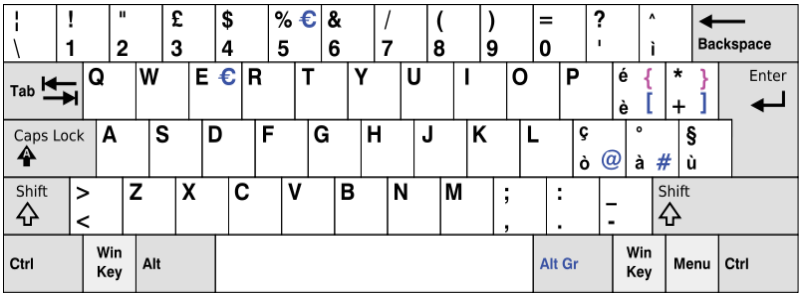

Figure 7. Italian Keyboard Layout (Source: http://wapedia.mobi/en/File:KB_Italian.svg). Reproduced from Wapedia under the Creative Commons Attribution/Share-Alike License and GNU Free Documentation License.

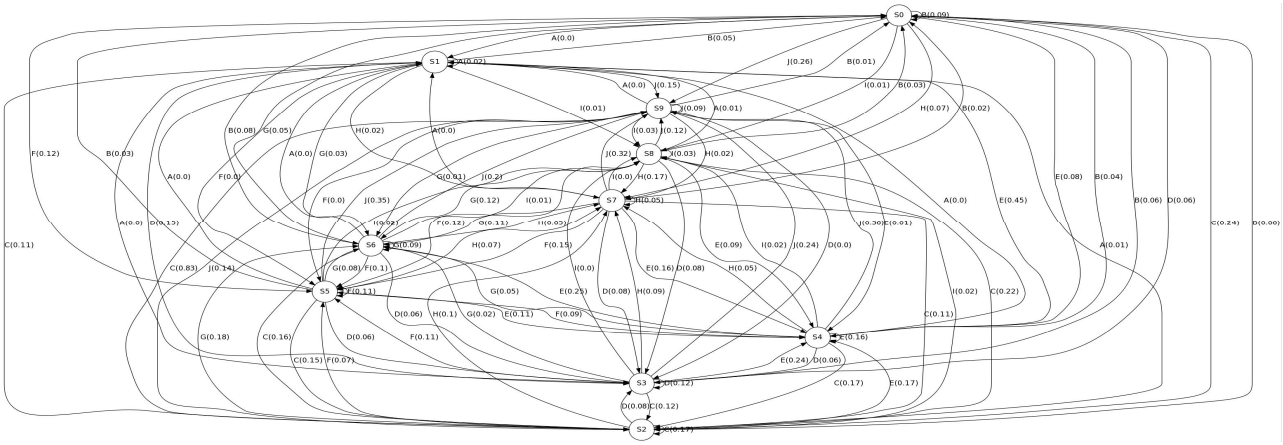

Figure 8. English HMM (10 states, 100 transitions)

histories of one letter.

Italian only had 4 clusters, which made it possible to achieve a statistically significant reconstruction with L = 3 and a similar volume of training data. Since there were a smaller number of possible transitions from each state, there were fewer low-probability state transitions. The training process had a larger sample set available for determining probability distributions. For Italian, the observed string increased to 4 symbols, meaning that conditional probability histories of up to 5 letters were considered.

Using window-size calculations from [6], we found the minimum string length needed to differentiate between the two models, with a 95% true-positive rate, was 77 symbols [29]. We therefore split the testing data into windows of 77 symbols and selected 400 windows. A set of 800 English and Italian windows were chosen to use for testing.

We used the test data to determine the ability of the English and Italian models to detect the language being used in interactive **SSH** sessions.

5. Detection Results

The testing data was sent through interactive SSH v2 connections following the keystroke pair delay distribu-

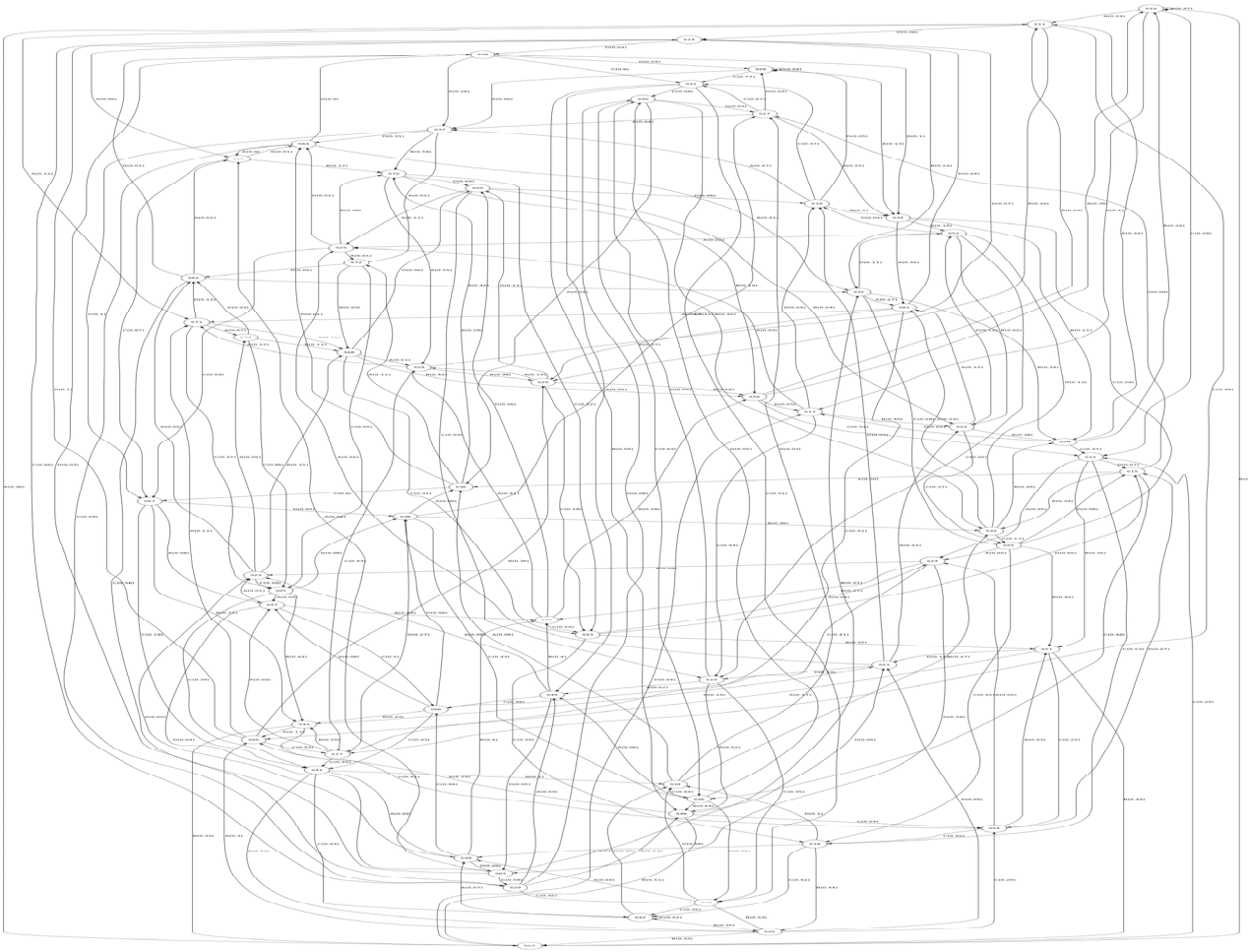

Figure 9. Italian HMM (64 states, 253 transitions).

tions described in Section 2.1. The detection procedure redirected the output of a parsed tshark[4] capture to a custom detection routine. The detection routine used our English and Italian HMMs with maximum likelihood and confidence interval detection criteria.

The ROC curves for the tests are shown in **Figures 10-13**:

- Circular points compare English and Italian data streams.
- Square points are for Malagasy (the national language of Madagascar) data streams, and
- Diamond points are for English (Italian) data transmitted with Italian (English) timing.

The latter tests were used to help clarify the relationship between timing and language letter sequence conditional probabilities.

From the plots denoted by circular markers in the ROC curves, where English and Italian are compared, it is clear both the CI and maximum likelihood approaches are able to detect the language used. It was found that with strings of 77 symbols, a threshold of 89.0% for using the HMM to detect English and 0.0% for using the HMM to detect Italian were optimal.

Since the range of Italian key-stroke delays is a subset of English, all English inputs produced impossible transitions within 77 symbols. The Italian conditional probabilities were consistent enough that Italian text never produced probabilities outside the 95% confidence interval. No impossible transitions occurred when Italian text was parsed by the English Markov Model. Also, the English conditional probabilities were less homogeneous. When English text was parsed it would often produce observed transition probabilities outside the 95% confidence interval. While this is to be expected approximately 5% of the time, our observations can be explained, in part, as an artifact of using the L = 1 approximation of the true process. However the conditional probabilities in Italian text were quite different from English conditional probabilities, explaining the optimal 89% threshold, as shown in **Table 2** this threshold was able to reliably differentiate between the two languages as shown in **Figure**

[4]http://sourceforge.net/projects/wireshark/

Table 2. ROC statistics for English vs. Italian (left) and Cross-Symbolization (right) with Confidence Intervals. The last column in the above tables, "Distance," is the distance from the curve at that point to the point (1, 0) on the axes. The ideal threshold is reached when this distance is minimized.

Threshold	True Pos	False Pos	True Neg	False Neg	Distance
0.00	401	401	0	0	1.000
Repeated 79 times					
0.80	401	401	0	0	1.000
0.81	401	392	9	0	0.978
0.82	401	371	30	0	0.925
0.83	401	294	107	0	0.733
0.84	401	201	200	0	0.501
0.85	401	103	298	0	0.257
0.86	401	40	361	0	0.100
0.87	401	9	392	0	0.022
0.88	399	3	398	2	0.009
0.89	**399**	**0**	**401**	**2**	**0.005**
0.90	397	0	401	4	0.010

Threshold	True Pos	False Pos	True Neg	False Neg	Distance
0.00	401	401	0	0	1.000
Repeated 78 times					
0.80	401	400	1	0	0.998
0.81	401	400	1	0	0.998
0.82	401	371	30	0	0.925
0.83	401	297	104	0	0.741
0.84	401	195	206	0	0.486
0.85	401	105	296	0	0.262
0.86	401	37	364	0	0.092
0.87	400	7	394	1	0.018
0.88	**395**	**0**	**401**	**6**	**0.015**
0.89	389	0	401	12	0.030

10.

In cross-symbolization, English was symbolized with the Italian delay statistics and the Italian symbol-space and vice-versa. This was done to see which of the two phases of our process (symbolization or HMM parsing) dominated the process. When the symbolizations were switched, the opposite language was identified. That is, for the English case, Italian was identified. From the results, shown by the plots with diamond markers in **Figures 10-13**, it was clear that the symbolization process dominated our approach. However, it wasn't clear if the behavior identified was a function solely of the symbolization or also due to language structure. We note that while English is a Germanic derivative language and Italian is a Latin derivative, both are in the Indo-European family and hence have substantial similarities.

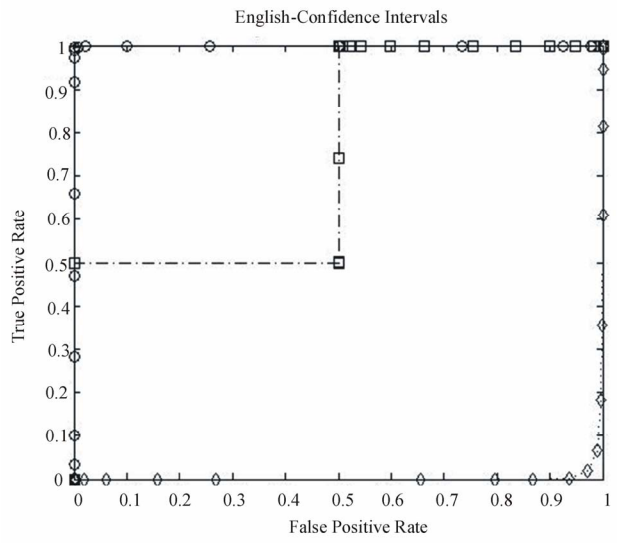

Figure. 10. English Confidence Interval results—English vs. Italian (circle), Cross-Symbolization (diamond), and Malagasy (square).

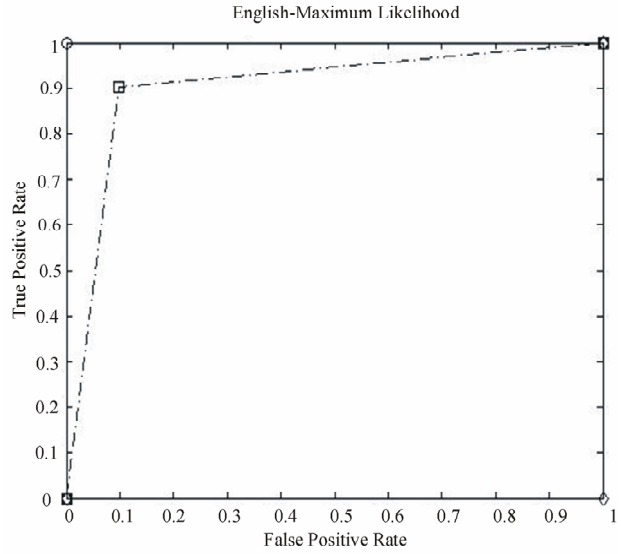

Figure. 11. English Maximum Likelihood results—English vs. Italian (circle), Cross-Symbolization (diamond), and Malagasy (square).

To address this issue, we compared the modern languages with ancestor languages. This experiment found

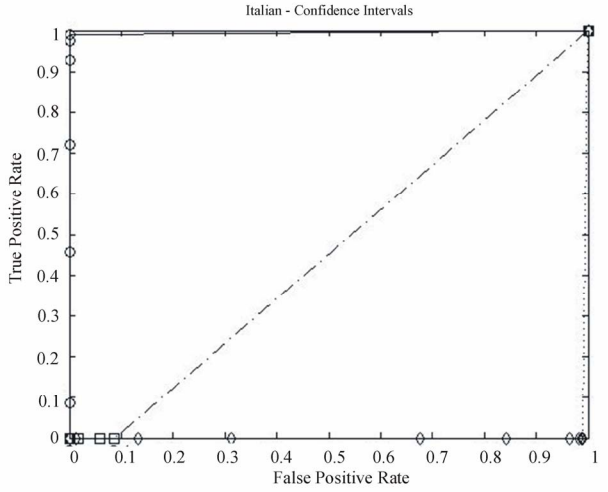

Figure. 12 Italian Confidence Interval results—English vs. Italian (circle), Cross-Symbolization (diamond), and Malagasy (square).

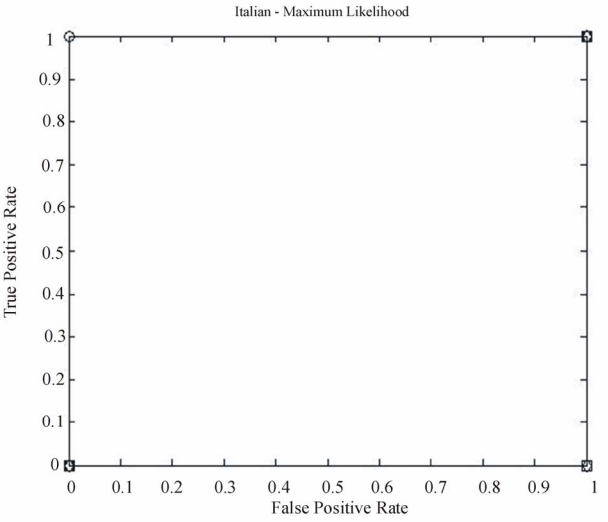

Figure. 13. Italian Maximum Likelihood results—English vs. Italian (circle), Cross-Symbolization (diamond), and Malagasy (square).

that when Old English[5] and Latin were symbolized like their younger languages, they were detected as their younger counterparts. It appears that related languages can be identified using our approach[6]. This further supported our hypothesis that detection was primarily based on symbolization. The final experiment used Malagasy, the national language of Madagascar. Malagasy was selected as it uses no diacritical marks, can be represented with the Latin character set, and does not originate from Sanskrit (as English and Italian do); it is a member of the Austronesian language family. The only text available in Malagasy was a copy of The Bible [35]. This test produced curious results. For this experiment, the book of Genesis was symbolized with both the English and Italian statistics and symbol-space. These symbolized strings are then analyzed using confidence intervals and maximum likelihood.

The results of these comparisons are presented in the plots with square markers in **Figures 10-13**. From these curves, it appeared that detection is strongly influenced by language structure and not solely symbolization as was indicated by the English-Italian cross-symbolization tests.

At first glance, it appeared that the ROC curves in **Figures 10-13** favored the existing maximum likelihood measure over the confidence interval approach proposed by Schwier. However, this was only because the strings presented are 77 symbols long. Floating-point underflow is possible with longer strings. While there are methods to avoid this, such as normalization at every step and the use of logarithms, these methods introduce more noise into an already noisy calculation.

It should also be noted that the confidence interval approach is for detection, not for classification as maximum likelihood is. CI methods indicate the presence of a specific behavior in a given sample string. It can be used as a classifier, but that isn't its intended use. As said earlier, to do so would require ROC curve inspection to determine a suitable threshold between behaviors.

In [6,28], Schwier *et al.* pointed out that with confidence intervals there is a marginally higher false positive rate. This is due to less noise being introduced than with maximum likelihood.

It should be noted that confidence interval analysis can be performed online. This is not the case for maximum likelihood testing. Furthermore, presenting the data in windows is necessary for online use [6], and for the differentiation between languages. While this windowing is not needed for maximum likelihood, it can be applied to it.

6. Conclusions and Future Work

Protocol tunneling is the basis of most protocol stacks. It can be used to add security and anonymity to networks by tunneling insecure protocols within other protocols. This may be used positively (ex. VPNs), or negatively (to circumvent network security measures). Side channel vulnerabilities are able to detect the use of protocol tunneling and in some cases counteract the reasons for using tunnels.

We provided a brief, but comprehensive survey of

[5]Beowulf

[6] This would assume that native speakers of Italian (English) would type Latin (Beowulf) with the same speed as their native language, which is likely not to hold in practice. Our test was run solely to see how sensitive this process was to the conditional probability structure of the language as opposed to typing dynamics.

known side channel vulnerabilities for protocol tunneling. We then provided a tutorial for our HMM approach to protocol detection. This approach is more general than the other approaches in the literature, since it is data driven. As an illustrative example, we show how language use can be detected in interactive SSH sessions. Our experimental results show that the example application was very successful.

6.1. Tor Analysis

In [36] we used the approach given in Sections 4 and 5 to trace network flows through Tor. While the details of that application are outside the scope of this paper, the results of that work are consistent with the survey given in Section 2. Our approach was passive and did not require a malicious Tor node. We found that, as with English, it was impossible to find the value of L for the Tor model. This was due to intermittent session reinitializations within Tor inserting large network delays that were not associated with the underlying protocol. We were able to construct a practical model that included only statistically significant states and transitions. Using this model and the Viterbi path traced by observed network streams, we were able to accurately classify 95% of the packets as belonging to the same network session. This was without requiring either additional network traffic or a global view of the network.

6.2. SSH Side-Channel Attacks

For language detection, we wrote a detector using Java. By redirecting the output of a parsed tshark capture to it, it is possible to detect the presence of English and Italian in real-time. To test the functionality of this application, the samples from Project Gutenberg were sent across an SSH tunnel to the client machine which was monitoring communication with the detector.

The test was successful: using a threshold of 0.0% with the Italian HMM and 89.0% with the English HMM it is possible to detect the presence of either language in a given sample string. That is, if the CI analysis shows that more than 89.0% of the behavior of the English HMM is exhibited by the string, it is English with a 5% false positive rate. This detection was performed in real-time and can be done from a third node as well as the packet contents were not needed, merely the delays between them.

6.3. Zero-knowledge HMM Detection of Protocol Tunnels

From the results of our tests, it is apparent that our HMM inference approach accurately detects complex tunneled applications. It is also possible to use these models to detect languages in real-time. Typing dynamics have been used as a form of biometrics for user identification/authentication for quite some time [5,31,32]. Our results show a possible broader application for this.

The symbolization phase is particularly important as it affects both the construction of the HMM as well as detection process. From the tests performed in this work, it is clear that symbolization affects the identification of behaviors present in a string. Given the number of key-pairs considered, it wasn't possible to symbolize based solely on the plots of the Gaussians approximation. Consequently, we used a clustering approach to identify centers of activity within the set of all delays. Furthermore, we recognized that as the symbol-space grows larger, the data required to build an HMM increases exponentially.

6.4. Future Work

Future work could look at increasing the efficiency of the recognition. The most important thing to note is that this analysis is made possible through the timing vulnerabilities present in most secure communication channels. As services seek to maintain a high quality of service, they attempt to minimize any introduced latency.

This is a major vulnerability and will likely be present for some time into the future [35]. The HMM inferencing approach we present is general and can be used to design tunnel detection routines for protocols that fulfill our assumptions. The stationary probability assumption is not very restrictive. Adding random noise to the probabilities simply produces a new probability distribution. Steadily decreasing delays would be problematic, except that a minimal delay will quickly be reached. Increasing delays uniformly would make the tunneled application unusable.

We are interested, however, in relaxing the finite state assumption. We are working at adapting this approach to probabilistic grammars. It would also be interesting to move further up the Chomsky hierarchy and look at probabilistic recursively enumerable processes.

7. Acknowledgements

This material is based upon work supported by, or in part by, the Air Force Office of Scientific Research contract/grant number FA9550-09-1-0173. Opinions expressed are those of the author and not the US Department of Defense. We thank the reviewer for their input, which improved the paper.

8. References

[1] J. Walrand and P. Varaiya, "High-Performance Commu-

nications Networks," Morgan-Kaufmann, San Francisco, 1996.

[2] O. Kolesnikov and B. Hatch, "Building Linux Virtual Private Networks (VPNs)," New Riders, Indianapolis, 2002.

[3] R. Craven, C. Abbott, H. Bhanu, J. Deng and R. R. Brooks, "Orwell was an Optimist," 6*th Annual Cyber Security and Information Intelligence Workshop*, Oak Ridge, 21-23 April 2010.

[4] M. Dusi, M. Crotti, F. Gringoli and L. Sagarelli, "Tunnel Hunter: Detecting Application-Layer Tunnels with Statistical Fingerprinting," *Communications Networks*, Vol. 53, No. 1, 2009, pp. 81-97. doi:10.1016/j.comnet.2008.09.010

[5] D. X. Song, D. Wagner and X. Tian, "Timing Analysis of Keystrokes and Timing Attacks on **SSH**," *SSYM*'01: *Proceedings of the* 10*th conference on USENIX Security Symposium*, Vol. 10, 2001, p. 25.

[6] J. Schwier, "Pattern Recognition for Command and Control Data Systems", Ph.D. Dissertation, ECE Department, Clemson University, Clemson, 2009.

[7] J. Schwier, R. R. Brooks, C. Griffin and S. Bukkapatnam, "Zero Knowledge Hidden Markov Model Inference," *Pattern Recognition Letters*, Vol. 30, No. 14, 2009, pp. 1273-1280. doi:10.1016/j.patrec.2009.06.008

[8] R. Dingledine, N. Mathewson and P. Syverson, "Deploying Low-Latency Anonymity: Design Challenges and Social Factors," *IEEE Security Privacy*, Vol. 5, No. 5, October 2007, pp. 83-87. doi:10.1109/MSP.2007.108

[9] R. Dingledine, "Current Events in Tor Development," 24*th Chaos Communication Congress* (24*C*3), Berlin, 27-30 December 2007.

[10] R. Craven, C. Abbot, H. Bhanu, J. Deng and R. R. Brooks, "Orwell Was an Optimist," *Cyber Security and Information Intelligence Research Workshop* 2010, Oak Ridge, 21-23 April 2010.

[11] N. Leavitt, "Anonymization Technology Takes a High Profile," *IEEE Computer*, Vol. 42, No. 11, 2009, pp. 15-18.

[12] D. Kaminsky, "Why We Were So Vulnerable to the DNS Vulnerability," 25*th Chaos Computer Congress*, Berlin, 17 January 2009. http://dewy.fem.tu-imenau..de/CCC/25C3/video_h264_720x756/25c3-2906-en-why_were_we_so_vulnerable_to_the_dns_vulnerability.mp4.torrent

[13] N. S. Evans, R. Dingledine and C. Grothoff, "A Practical Congestion Attack on Tor Using Long Paths," 18*th USENIX Security Symposium*, Berkeley, 2009.

[14] A. Hintz, "Fingerprinting Websites Using Traffic Analysis," *Proceedings of the Workshop on Privacy Enhancing Technologies* 2002, Berlin, 10 May 2002.

[15] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose and G. M. Masson, "Uncovering Spoken Phrases in Encrypted Voice over IP Conversations," *ACM Transactions on Information and Systems Security*, Vol. 13, No. 4, 2010, pp. 35:1-35:30.

[16] Y. Zhu, X. Fu, R. Bettatli and W. Zhao, "Anonymity Analysis of Mix Networks Against Flow Correlation Attacks," *Proceedings IEEE Global Communications Conference* (*GLOBECOM*), College Station, 28 Novenber-2 December 2005

[17] Y. Zhu, X. Fu, B. Graham, R. Bettati and W. Zhao, "Correlation-Based Traffic Analysis Attacks on Anonymity Networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 21, No. 7, May 2010, pp. 954-967. doi:10.1109/TPDS.2009.146

[18] S. J. Murdoch and P. Zielinski, "Sampled Traffic Analysis by Internet-Exchange-Level Adversaries," *Privacy Enhancing Technologies LNCS*, Springer, Berlin, 2007. doi:10.1007/978-3-540-75551-7_11

[19] Y. Guan, X. Fu, D. Xuan, P. U. Shenoy, R. Bettati and W. Zhao, "Netcamo: Camouflaging Network Traffic for QoS-Guaranteed Mission Critical Applications," *IEEE Transactions on Systems, Man, and Cybernetics*: *Part A*: *Systems and Humans*, Vol. 31, No. 4, July 2001, pp. 253-265. doi:10.1109/3468.935042

[20] L. Overlier and P. Syverson, "Locating Hidden Servers," *IEEE Symposium on Security and Privacy*, No. 1, 2006, pp. 100-114.

[21] S. Murdoch and G. Denezis, "Low-Cost Traffic Analysis of Tor," 2005 *IEEE Symposium on Security and Privacy*, Oakland, 8-11 May 2005.

[22] L. Xin and W. Neng, "Design Improvement for Tor Against Low-Cost Traffic Attack and Low-Resource Routing Attack," 2009 *WRI International Conference on Communications and Mobile Computing*, Vol. 3, January 2009, pp. 549-554. doi:10.1109/CMC.2009.18

[23] R. Wiangsripanawan, W. Susilo and R. Safavi-Naini, "Design Principles for Low Latency Anonymous Network Systems Secure against Timing Attacks," *ACSW'07 Proceedings of the* 5*th Australasian Symposium on ACSW Frontiers*, Vol. 68, 2007, pp. 183-191.

[24] S. J. Murdoch, "Hot or Not: Revealing Hidden Services by their Clock Skew," *Proceedings of the* 13*th ACM conference on Computer and Communications Security*, *CCS* 06, Alexandria, 30 October-3 November 2006, pp. 27-36.

[25] S. Zander and S. J. Murdoch, "An Improved Clock-Skew Measurement Technique for Revealing Hidden Services," *SS*'08 *Proceedings of the* 17*th conference on Security Symposium*, San Jose, 28-30 April 2008, pp. 211-225.

[26] C. R. Shalizi, K. L. Shalizi and J. P. Crutchfield, "An Algorithm for Pattern Discovery in Time Series," *The Computing Research Repository*, October 2002. cs.LG/021005: http://arxiv.org/abs/cs.LG/021005.

[27] R. R. Brooks, J. M. Schwier and C. Griffin, "Behavior Detection Using Confidence Intervals of Hidden Markov Models," *IEEE Transactions on SMC Part B*, Vol. 39, No. 6, 2009, pp. 1484-1492.

[28] N. Hopper, E. Y. Vasserman and E. Chan-Tin, "How Much Anonymity Does Network Latency Leak," *ACM Transactions on ACM Transactions on Information and System Security* (*TISSEC*), Vol. 13, No. 2, 2010, pp. 13:1-13:28.

[29] J. Schwier, R. R. Brooks and C. Griffin, "Methods to

Window Data to Differentiate between Markov Models," *IEEE Transactions on System Man and Cybernetics*, *Part B*: *Cybernetics*, Vol. 41, No. 3, 2010, pp. 650-663. doi:10.1109/TSMCB.2010.2076325

[30] J, Schwier, "Pattern Recognition for Command and Control Data Systems," PhD Dissertation, Holcombe Department of Electrical and Computer Engineering, Clemson University, Clemson, July 2009.

[31] K. Hempstalk, "Continious Typist Verification Using Machine Learning," Ph.D. Dissertation, Department of Computer Science, University of Waikato, Hamilton, 2009

[32] D. Gunetti and C. Picardi, "Keystroke Analysis of Free Text," *ACM Transactions on Information and System Security*, Vol. 8, No. 3, 2005, pp. 312-347. doi:10.1145/1085126.1085129

[33] B. Fritzke, "Fast Learning with Incremental RBF Networks," *Neural Processing Letters*, Vol. 1, No. 1, 1994, pp. 2-5. doi:10.1007/BF02312392

[34] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, Vol. 77, No. 2, 1989, pp. 257-286. doi:10.1109/5.18626

[35] http://www.madabibliq.org/ (last visited May 2010).

[36] R. Craven, "Traffic Analysis of Anonymity Systems," MS Thesis, Holcombe Department of Electrical and Computer Engineering, Clemson University, Clemson, May 2010.

