PURDUE UNIVERSITY TECHNICAL REPORT

Camouflaging Timing Channels in Web Traffic

Sarah H. Sellke, Chih-Chun Wang, Saurabh Bagchi
School of Electrical and Computer Engineering
465 Northwestern Ave.
Purdue University
West Lafayette, IN 47907-2035
{ssellke,chihw,sbagchi}@purdue.edu

Ness B. Shroff
Departments of CS and ECE
2015 Neil Avenue
The Ohio State University
Columbus, OH 43210
shroff@ece.osu.edu

Abstract

Web traffic accounts for more than half of Internet traffic today. Camouflaging covert timing channels in Web traffic would be advantageous for concealment. In this paper, we investigate the possibility of disguising network covert timing channels as HTTP traffic to avoid detection. Extensive research has shown that Internet traffic, including HTTP traffic, exhibits self-similarity and long range persistence. Existing covert timing channels that mimic i.i.d. legitimate traffic cannot imitate HTTP traffic because these covert traffic patterns are not long range dependent. The goal of this work is to design a covert timing channel that can be camouflaged as HTTP traffic. To this end, we design a covert timing channel whose inter-arrival times are long range dependent and have the same marginal distribution as the inter-arrival times for new HTTP connection traffic. These inter-arrival times are constructed by combining a Fractional Auto-Regressive Integrated Moving Average (FARIMA) time series and an i.i.d. cryptographically secure random sequence. Experiments are conducted on PlanetLab, and the results are validated against recent real traffic trace data. Our experiments demonstrate that the traffic from this timing channel traffic is statistically indistinguishable from legitimate HTTP traffic and undetectable by all current detection schemes for timing channels.

1. Introduction

Due to the rapid growth of Internet and Web based applications, covert communication over the Internet has received increased attention from industry and the research community. Since traditional firewalls do not usually exam packet inter-arrival times, covert timing channels could be an attractive way for confidential communication between untrusted systems. With the emergence of new designs of network timing channels, detection methods attempt to differentiate these timing channels from legitimate traffic. The question that we ask us is whether we design network timing channels that behave like legitimate traffic and that are robust to the timing noise characteristic on the Internet.

In our prior work, we have presented a computationally non-detectable timing channel for mimicking legitimate *i.i.d.* traffic [5] . The marginal distribution of the packet inter arrival times from this timing channel can be any probability distribution. However, these *i.i.d.* timing channels cannot mimic traffic that is auto-correlated.

Extensive research has shown that aggregated traffic is self similar and long range dependent (LRD). For instance, the TCP connection start time for HTTP traffic, the most observed traffic on the Internet [21], is shown to be LRD and non-stationary [10]. Our goal is to design a covert timing channel that can mimic LRD traffic, such as HTTP traffic. In particular, we would like our timing channel traffic not only to have the desired marginal distribution, but the same autocorrelation function (equivalently, same long range dependence) as legitimate traffic trace data.

To achieve this goal, we first analyze traffic trace data for HTTP traffic from 2009 available from CAIDA [1]. We use the insights from these traffic traces to revive and expand a prior statistical model for HTTP traffic [10] to better fit current data. Our model uses a Fractional Auto-Regressive Integrated Moving Average (FARIMA) time series to capture LRD behavior and also matches the marginal distribution of the data. We then create covert timing channel traffic by embedding covert information in this model without disturbing any first- or second-order statistical behavior of the legitimate traffic. We implement the LRD timing channel and conduct experiments with geographically distributed senders and receivers on PlanetLab. Our experimental results indicate that the new timing channel traffic is statistically indistinguishable from legitimate traffic, and our tests confirm it evades the best available detection methods.

One scenario of using this timing channel is when a sender resides in a country with tight censorship. She is able to manipulate the inter-transmission times of aggregated HTTP traffic within her organization to communicate in a covert manner with a receiver who is outside the geographical boundary of the country and where there are no such censorship laws. Since HTTP traffic has a high volume in most settings, the sender can achieve reasonable rates of covert communication. The receiver intercepts the traffic en-route to the final web servers, observes the inter-reception times, and decodes the privileged information from them using a code-book that the sender and the receiver have agreed to *a priori*.

The remainder of our paper is organized as follows: In Section 2, we review related work on network timing channels and long range dependent traffic. In Section 3, we present our analysis on CAIDA data and updated traffic models. In Section 4, we describe our design and implementation of covert HTTP timing channels. Our experimental results are described in Section 5. We conclude with discussion and future research directions in Section 6.

[1]Support for CAIDA's Internet traces is provided by the National Science Foundation, the US Department of Homeland Security, and CAIDA Members.

2. Related Work

Past research on network timing channels has investigated channel capacity, schemes for reducing the capacity, designs of network timing channels, and detection schemes to identify the existence and usage of timing channels.

Early implementation of an on/off timing channel [1] demonstrated the feasibility of leaking information by a network covert timing channel. In [4], the authors built a Keyboard JitterBug, a device interposed between the keyboard and the computer, that can leak typed information through a covert network timing channel when a user runs an interactive application such as *ssh*. We designed a timing channel [5] that maps L-bits strings to n packet inter transmission times.This design includes both the on-off scheme and the keyboard jitter bugs as special cases. It significantly improved the data rate of the timing channel. In fact, the data rate of this scheme is close to the theoretical upper bound – the achievable rate of the geometric codes.

Given the threat of clandestinely leaking information by network covert timing channels, researchers found ways to detect them [1, 2, 3]. However, network timing channels can be very surreptitious. In [5], we constructed a computationally non-detectable timing channel, mimicking any *i.i.d.* legitimate traffic patterns. The inter-transmission times from telnet traffic are shown to be *i.i.d.* from a Pareto distribution [16]. When used to imitate telnet traffic, our timing channel can evade detections entirely. In spite of the strong non-detectability property, the usage of this timing channel is limited to imitating *i.i.d.* legitimate traffic. It cannot be used to imitate correlated traffic such as HTTP traffic, which represents more than 50% of Internet traffic today. It is well-established that HTTP traffic is non-stationary and long range dependent (LRD) [10, 11]. LRD means the autocorrelations are positive and decay slowly, thus are not summable. The focus of this work is to construct timing channels that emulate the long range dependence of HTTP traffic.

The discovery of long range dependence and self-similarity of Internet traffic ([13]) had profound effect on our understanding of network traffic. Many papers on long range dependent and self similar network traffic have been published. Readers are referred to [12] for a comprehensive overview.

Mathematical models and simulations are essential for network performance evaluation, traffic controls, and resource provisioning. Synthesized long range dependent traffic is required as an input process for simulations in order to reflect the reality that many traffic variables are long-range persistent. Fractional Gaussian noise (FGN) and fractional ARIMA (FARIMA) processes are the two most widely-used input processes for network simulations. Cleveland *et al.* proposed a stochastic model well suited for TCP start time for HTTP traffic [10]. The authors analyzed 23 million TCP connections organized into 10704 blocks of approximately 15 minutes each. They used a FARIMA sequence to capture the long range persistence, and the model produces synthetic traffic stochastically similar to that from the actual wire of an Internet link.

A fast Fourier transform method for synthesizing approximate sample paths for Fractional Gaussian Noise (FGN) is proposed in [17]. A summary of other methods for generating realistic network traffic can be found in [17].

Synthesizing a realistic traffic trace is evidently a crucial part of constructing timing channel traffic to evade detection. Unlike traditional modeling and synthetic trace generation, the synthetic traffic for our timing channel must meet more rigorous requirements: 1) be statistically indistinguishable (not just similar) from the underlying HTTP traffic to avoid detection; and 2) be able to transmit covert information through timing, and of course, be decodable by the receiver.

Next, we will present a stochastic model for HTTP new connection inter arrival times based on packets trace data collected by CAIDA in 2009.

3. Model for A LRD Timing Channel

The goal of this research is to design a network timing channel that can be hidden within HTTP traffic. *Persistent connections* are adopted in HTTP/1.1, that is, a single TCP connection is created and reused for multiple HTTP request/response interactions. Our timing channel will use the new TCP connection inter-arrival times, not HTTP packet inter-arrival times, to carry covert information. Due to the network jitters distorting timing information, larger inter-arrival times are more resilient to decoding errors. Therefore, the TCP connection inter-arrival times, larger than HTTP packet arrival times, are more reliable for transmitting covert information. The tradeoff of the enhanced robustness is the corresponding throughput reduction when compared to the packet inter-arrival time schemes in [1, 4, 5].

We use the packet traces collected by CAIDA in March 2009 as a baseline for comparing our covert timing traffic and legitimate traffic. This dataset contains anonymized passive traffic traces from CAIDA's Equinix-Chicago and Equinix-Sanjose monitors on OC192 Internet backbone links. The Equinix-Chicago Internet data collection monitor is located at an Equinix datacenter in Chicago, IL, and is connected to an OC192 backbone link (9953 Mbps) of a Tier1 ISP between Chicago, IL and Seattle, WA.

The original data set from Equinix-Chicago direction A contains approximately 15 GB of compressed data. We first extract the new TCP connection packets for HTTP from the dataset. The resulting data is only about 1% of the original data. Since the attack scenario under consideration is a compromised edge router of an enterprise network leaking information via timing channels, we further partition the new TCP connection packet trace into subnets according to their 8 bits network prefix. Within each subnet, we divide the one-hour trace into four 15-minutes intervals as in [10].

3.1. Limitation of the Existing Model for HTTP Traffic

A statistical model [10] for TCP new connection times was developed, based on traffic traces collected at Bell Labs between 1998 and 2000. The authors conducted extensive empirical as well as in-depth theoretical studies of 23 million TCP connections collected at Bell Labs between 1998 and 2000. They concluded that TCP start times for HTTP are nonstationary and LRD, the marginal distribution of the inter-arrival times is approximately Weibull, and the autocorrelation of the log inter-arrival times is modeled by adding white noise to a FARIMA time series.

FARIMA models are generalizations of *Autoregressive Integrated Moving Average* ARMA model by allowing fractional values d in the degree of difference [15]. It is commonly used to model long range dependent behavior. The FARIMA series s_j can be generated from equation (1):

$$(I-B)^d s_j = \epsilon_j + \epsilon_{j-1} \tag{1}$$

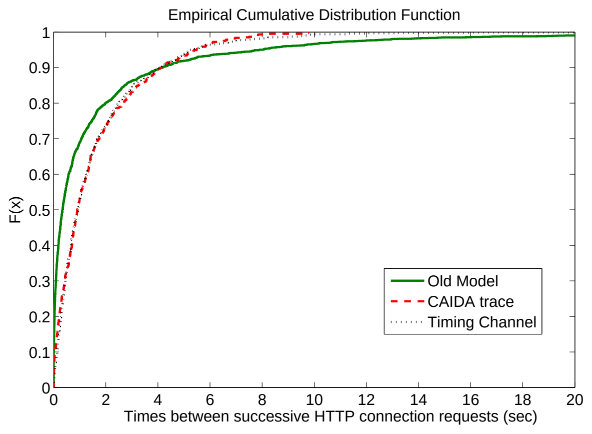
where B is the backward shift operator ($Bs_j = s_{j-1}$) and ϵ_i are *i.i.d.* Gaussian random variables with mean 0 and variance σ_ϵ^2. Their model is developed for data with Hurst parameters around 0.75, and they used a fixed value 0.25 for the degree of difference d. The Hurst parameters for the Bell Lab data are approximately 0.75, and the relationship between H and d is $H = d + 0.5$.

One advantage of this model is that only one parameter, r, the rate of new TCP connection arrival times, is needed to generate the traffic trace. In their model, the parameters (α and λ) for a Weibull distribution are functions of the connection rate r. The corresponding *c.d.f.* of the marginal distribution of the new connection inter-arrival times for is:

$$F(t) = 1 - e^{-(t/\alpha(r))^{\lambda(r)}}, t \geq 0 \tag{2}$$

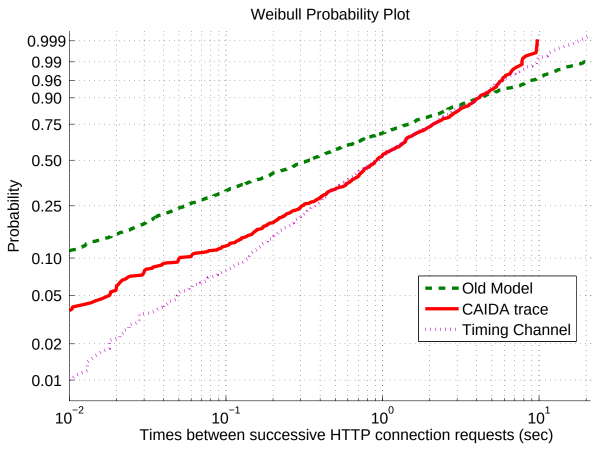
The Internet has changed tremendously in the last decade. It is not surprising that this model does not fit current CAIDA data well. For instance, we select a dataset consisting of a 15-minute block of CAIDA traffic trace from a subnet, and compare it with the model in [10]. The load is calculated as $r = 0.6387$ connection/seconds for this data. We then create a synthetic dataset according to the model in [10] using $r = 0.6387c/s$.

Figure 1(a) compares the empirical cumulative distribution functions from the CAIDA trace and the trace generated according to the model in [10]. Although it may appear that their empirical *cdfs* are very close to each other, the two-sample Kolmogorov-Smirnov test rejects the null hypothesis that these two samples are drawn from the same distribution at level 0.05. The maximum distance between the two empirical distributions is 0.2558, and the p-value is $2.8 \cdot 10^{-27}$. This conclusion is further confirmed visually by the Weibull plots in Figure 2(a). While the marginal distribution of CAIDA's new TCP connection times is approximately Weibull, it does not come from the same Weibull distribution as the model in [10].



(a)

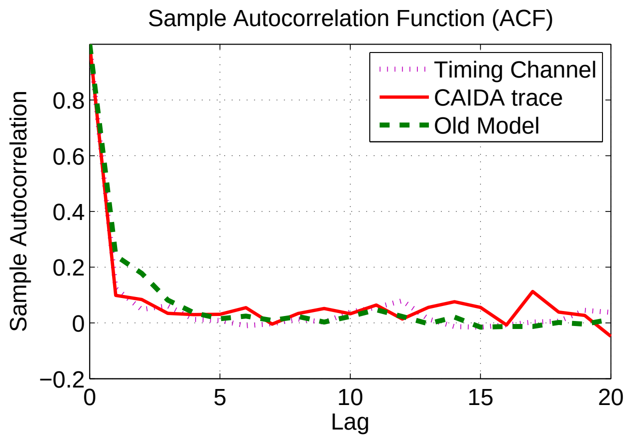
Figure 1. Comparison of Empirical cdf's



(a)

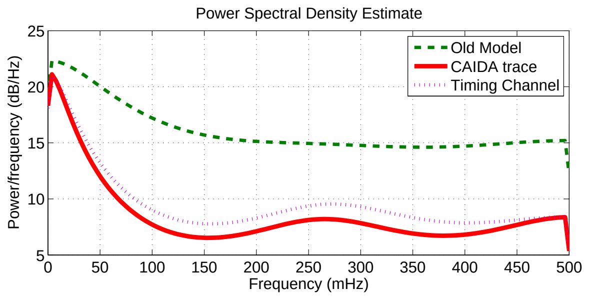
Figure 2. Fragment of Weibull Plots

In addition to the mismatch of the marginal distribution between the CAIDA data and the model, the second order statistics from the two data sets also differ. The Hurst parameter for the log inter-arrival times in the CAIDA data set is 0.61, while the model is developed for data with Hurst parameters around 0.75. In the existing model, no calculation of the Hurst parameter value is done based on the data. Instead, the value of the Hurst parameter is fixed as 0.75. Another misfit is the autocorrelation function. Since it is difficult to discern the differences from the autocorrelation plots in Figure 3(a), we plot the power spectrum density (PSD) estimates in Figure 4(a). It shows significant difference between the PSD estimates of the CAIDA data and of model in [10]. The PSD and the ACF of a time series form a Fouries transform pair, i.e. the PSD is the Fourier transform of the autocorrelation function of the time series, assuming the time series is wide sense stationary. The difference in PSD suggests different second order statistics.



(a)

Figure 3. Sample Autocorrelation Functions



(a)

Figure 4. Power Spectrum Density Estimates

The discrepancy described above indicates the need of updating the model for HTTP traffic in order to create timing channels that can hide in today's HTTP traffic.

3.2. New Model for HTTP Traffic

We first design a new model that expands the model in [10], so that it can model traffic with Hurst parameters other than 0.75. In our model, the scale and shape parameters for Weibull distribution, denoted by α and λ respectively, are estimated directly from the data, not computed from the load parameter r, to better fit a particular trace data.

Table 1 contains the notations we use in our model[2]. The TCP new connection inter-arrival times are denoted as $t_j, j = 1, \cdots, n$. Since t_j can vary by several orders of magnitude, $l_j = \log_2(t_j)$ is used for model fitting as in [10]. The marginal distribution of the $t_i's$ is approximately Weibull, with shape parameter $\lambda > 0$ and scale parameter $\alpha > 0$. Its cumulative distribution function (*c.d.f.*) is then:

$$F(t) = 1 - e^{-(t/\alpha)^{\lambda}}, t \geq 0; \alpha, \lambda > 0 \tag{3}$$

The pseudocode for generating synthetic traces using our model is shown in *Algorithm 3.1 NewModel*, There are four *input* parameters in our synthetic trace generation model: α, λ, H, and ρ_1. Here, α and λ are the scale and shape parameters of the Weibull distribution, H is the Hurst parameter of $l_j = \log_2(t_j)$, and ρ_1 is the autocorrelation of l_j at lag 1. The values of all four parameters are estimated directly from the data trace.

In the first 10 steps, we calculate all the parameters needed for synthetic trace generation. We will show the derivation of the values of these parameters shortly as we build our model. Steps 11 to 15 in *for loop* are the main component for generating n data points.

The *output* of the algorithm is simply an array that contains the sequence of inter-arrival times. They are statistically indistinguishable from today's HTTP traffic. As shown in Figures 1 to 4, the inter-arrival times from our new model match the marginal distribution and second order statistics of the real traffic

[2]we use the same notation as [10] whenever possible in our new model

trace. This new model will later be used to generate covert timing channel traffic, which will be shown in Section 4.

The first step, $d = H - 0.5$ calculates the degree of difference in the FARIMA model. The Euler constant $\gamma = 0.5772$ is set in step 2, and it is used in step 3 for calculating the mean of $l_j = \log_2(t_j)$. The variance of l_j is calculated in step 4 using the Weibull shape parameter λ. The variance of s_j (denoted as σ_s^2) is calculated in step 5, and it is used in step 11 for FARIMA series generation. The value of σ_n^2 is calculated in step 6, and is used, along with the value of μ_l from step 3, to generate an *i.i.d.* Log-Weibull sequence in step 12. We will show how the formulas for σ_s^2 and σ_n^2 are developed shortly. The step 7 calculates the load r of the TCP new connections as $r = 1/E[t_j]$, and $E[t_j] = \Gamma(\alpha(1+1/\lambda))$. Note, $\Gamma(\cdot)$ is the Gamma function defined as: $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$. The load r is then used in steps 8 through 9 to calculate parameters b_0, b_1, and b_2, which are used to obtain l_j from v_j in step 14.

Algorithm 3.1: NEWMODEL$(\alpha, \lambda, H, \rho_1)$

```
[1]$d = H - 0.5$
[2]$\gamma \leftarrow 0.5772$        //Euler Constant
[3]$\mu_l = \log_2(\alpha) - \gamma \log_2(e)/\lambda,$
[4]$\sigma_l^2 = \pi^2 \log_2^2(e)/6\lambda^2$
[5]$\sigma_s^2 = \sigma_l^2 \rho_1 (2-d)/(1+d)$
[6]$\sigma_n^2 = \sigma_l^2 - \sigma_s^2$
[7]$r = 1/(\alpha\Gamma(1+1/\lambda))$
[8]$b_0 = 0.7 - e^{-0.7088-0.05857r}$
[9]$b_1 = 1 - e^{-1.6301-0.06399r}$
[10]$b_2 = -e^{-4.1896-0.06254r}$
for $j \leftarrow 1$ to $n$
    do  [11] $s[j] \leftarrow$ FARIMA sequence with variance $\sigma_s^2$
        [12] $n[j] \leftarrow i.i.d.$Log-Weibull $(\mu_l, \sigma_n^2)$ sequence
        [13] $v[j] = s[j] + n[j]$
        [14] $l[j] = b_0 + b_1 v[j] + b_2 v^2[j]$
        [15] $t[j] = 2^{l[j]}$
return $(t)$
```

We now will explain the two key components, the FARIMA sequence (step 11) and the *i.i.d.* random sequence with a Log-Weibull distribution [3] (step 12) in our algorithm. We will show how we use these two sequences to build a model that matches the first and second order statistics of a data trace.

A FARIMA series is commonly used to model long range dependent behavior [15]. It is a generalization of ARMA model by allowing fractional values d in the degree of difference. The LRD series s_j can be generated from equation (1):

$$(I - B)^d s_j = \epsilon_j + \epsilon_{j-1} \tag{4}$$

where B is the backward shift operator ($Bs_j = s_{j-1}$) and ϵ_i are *i.i.d.* Gaussian random variables with mean 0 and variance σ_ϵ^2.

[3]Log-Weibul is a type of extreme-value distributions.

Table 1. List of Notations

Symbol	Explaination
t_j	HTTP connection inter-arrival times
α, λ	Weibull scale and shape parameters
$r = 1/E[t_j]$	new TCP connection rate
l_j	log scale of t_j : $l_j = \log_2(t_j)$
ρ_1	autocorrelation of l_j at lag one
H	Hurst Parameter of l_j
d	degree of difference in FARIMA model $d = H - 0.5$
s_j	FARIMA series: $(I - B)^d s_j = \epsilon_j + \epsilon_{j-1}$ ϵ_j are *i.i.d.* Gaussian$(0, \sigma_\epsilon^2)$
n_j	*i.i.d.* Log-Weibull (μ_l, σ_n^2) random variables uncorrelated with s_j
v_j	intermediate random sequence to model l_j: $v_j = s_j + n_j$

The long range dependence property of t_j is well modeled by Eq (1) [10, 15]. The advantage of FARIMA models is that it can capture the LRD using only one parameter – Hurst Parameter H. The degree of difference d is $d = H - 0.5$. In the earlier model [10], d is fixed to 0.25; That model fitted the Bell Lab data well since the Hurst parameters from those data are approximately 0.75. When a traffic trace has a significantly lower or higher Hurst parameter, like recent CAIDA data, the old model is no longer appropriate. One of our contribution is to model LRD traffic with wide range of Hurst parameters, we allow d to be in $(0, 0.5)$, corresponding to Hurst parameters in $(0.5, 1)$.

The *i.i.d.* Log-Weibull sequence $\{n_j\}$ is added to s_j, in an attempt to capture the first order and second order statistics of l_j. This method was first proposed in [10]. The random sequence $\{n_j\}$ and $\{s_j\}$ are not correlated, i.e. $cov(n_i, s_j) = 0$ for all i, j. Recall that $l_j = \log_2(t_j)$, and t_j is Weibull(α, λ). Thus l_j has a Log-Weibull distribution with mean μ_l and variance σ_l^2. The values of μ_l and σ_l^2 can be expressed in terms of the Weibull parameters λ and α: $\mu_l = \log_2(\alpha) - \gamma \log_2(e)/\lambda$, and $\sigma_l^2 = \pi^2 \log_2^2(e)/6\lambda^2$.

Our goal is to obtain l_j from s_j and n_j. For the ease of expositions, we use an intermediate variable v_j, and denote $v_j = s_j + n_j$. The goal is to have v_j statistically as close to l_j as possible. Thus, we design s_j and n_j, so that v_j satisfies $E[v_j] = \mu_l$ and $var[v_j] = \sigma_l^2$. In addition, v_j retains the same second order statistics of l_j. The Hurst parameter H and the autocorrelation of l_j at lag one, ρ_1, are obtained from the data.

We calculate the autocorrelation function $a_s(k)$ for s_i according to [15], and obtain:

$$a_s(k) = a_x(k) \cdot \frac{2k^2(1-d) - (1-d)^2}{k^2 - (1-d)^2}$$

where

$$a_x(k) = \frac{d(1+d)\cdots(k-1+d)}{(1-d)(2-d)\cdots(k-d)}$$

In particular, the autocorrelation of s_j at lag one is

$$a_s(1) = \frac{1+d}{2-d}$$

Since $v_j = s_j + n_j$ and $var(v_j) = \sigma_l^2$, we have

$$\sigma_l^2 = \sigma_s^2 + \sigma_n^2$$

Define

$$\theta = \frac{\rho_1}{a_s(1)} = \rho_1 \frac{2-d}{1+d},$$

where ρ_1 is the autocorrelation of l_j at lag one, an input parameter to our new model. Then, $\theta = \sigma_s^2/\sigma_l^2$, so that $\sigma_s^2 = \theta\sigma_l^2$.

Also we obtain the value of the variance of s_j using results in [15],

$$\sigma_s^2 = \frac{2}{1-d} \cdot \frac{\Gamma(1-2d)}{\Gamma^2(1-d)} \cdot \sigma_\epsilon^2$$

so that,

$$\sigma_\epsilon^2 = \frac{\theta(1-d)}{2} \cdot \frac{\Gamma^2(1-d)}{\Gamma(1-2d)} \cdot \sigma_l^2$$

The value of σ_ϵ^2 is used for generating the FARIMA sequence s_j defined by equation (1), which is used in step 11 in our model.

The parameters μ_l and σ_n^2 are used for generating *i.i.d.* Log-Weibull random sequence n_j in step 12, where

$$\sigma_n^2 = (1-\theta) \cdot \sigma_l^2$$

Even though v_j has the desired second order statistics, and satisfies $E(v_j) = \mu_l$ and $var(v_j) = \sigma_l^2$, its marginal distribution is not the desired Log-Weibull distribution. Therefore, to obtain l_j from v_j with the desired Log-Weibull distribution, we apply the following transformation:

$$l_j = b_0(r) + b_1(r)v_j + b_2(r)v_j^2$$

is used to obtain l_j in step 14, so that l_j has the desired Log-Weibull distribution. The values of b_0, b_1, and b_2 are computed in steps 8 to 10, according to equations (17), (18), and (19) on page 168 of [10], We made some adjustment to b_0 for a better fit. Note that b_0, b_1, and b_2 incoporate the effect of load on the inter-arrival times.

We generated synthetic traces according to our new model, and compare them with the real data and the existing model in [10]. The input parameters for our new model are estimated directly from the real data set. The value of the Hurst parameter H is estimated using R/S method [4]. Other methods and tools for Hurst parameter estimation can be found in [19]. The value of ρ_1 is estimated using a *Matlab* function *autocorr*. The Weibull parameters (α, λ) are estimated using the *Matlab* function *wblfit*. The CAIDA data set we used to generate figures 1(a) to 4(a) has the following parameter values: $\alpha = 1.36, \lambda = 0.76, H = 0.65, \rho_1 = 0.18$. The load r is 0.64 c/s.

[4]R/S method is also known as the rescaled adjusted range statistics method.

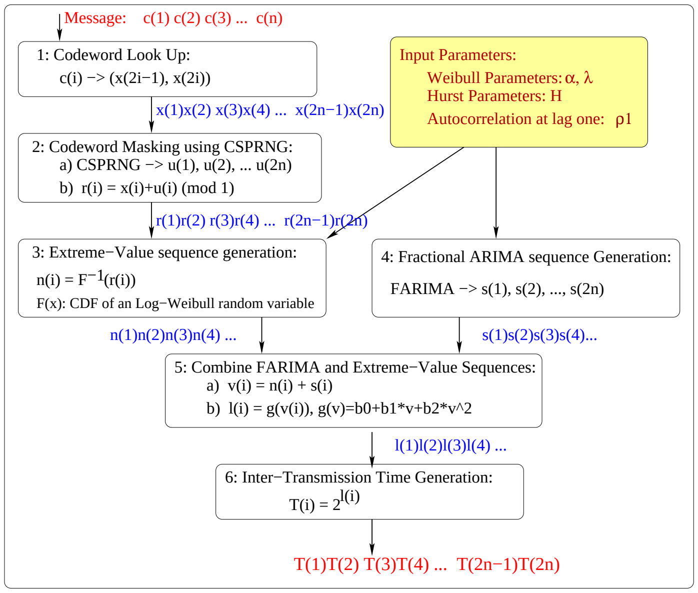
Figure 1(a) compares the empirical cumulative distribution functions from our model, the model in [10] and the CAIDA trace. In this figure, the empirical *cdf* of data from our model almost follow that of the CAIDA trace exactly. The Weibull plots in Figure 2(a) are also very close between our model and the data. There two figures demonstrate that the marginal distribution from our model is a much closer match to the real data than the existing model.

The second order statistics also match well between our model and the data. Autocorrelation plots are in Figure 3(a), and the power spectrum density (PSD) estimates are in Figure 4(a). Figure 4(a) shows the PSD estimates of the CAIDA data is much closer to that of our model than that of the existing model. These two figures show that the second order statistics from our model also matches the real data better than the existing model.

The fundamental reasons behind the better match is that we use H and ρ_1 in our model, in addition to the load(r) and that we measured these values directly from recent real data.

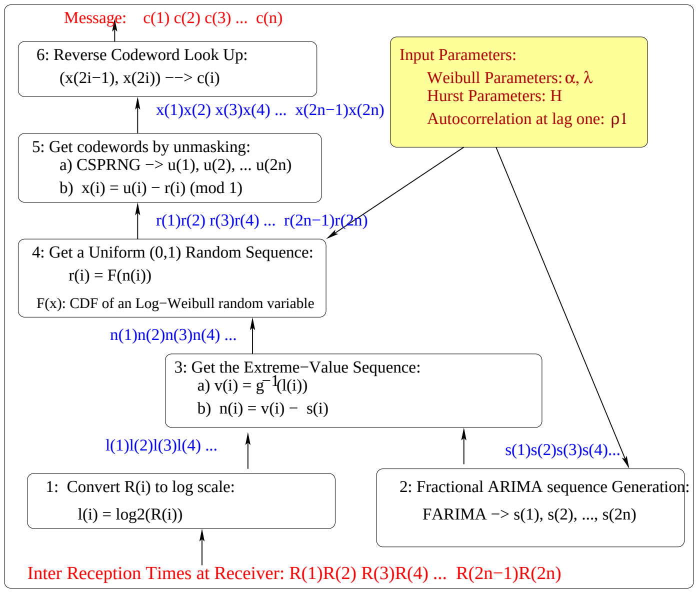
4. Design of HTTP Timing Channel

As we have seen, our model can be used to generate synthetic data that is statistically indistinguishable from the real trace. If we can embed covert information in our model while maintaining the statistical properties, a detection-resistant covert timing channel can be created. In what follows, we will explain how we incorporate this model in our HTTP timing channel design.



Message: c(1) c(2) c(3) ... c(n)
1: Codeword Look Up:
c(i) -> (x(2i−1), x(2i))
x(1)x(2) x(3)x(4) ... x(2n−1)x(2n)
Input Parameters:
Weibull Parameters: α, λ
Hurst Parameters: H
Autocorrelation at lag one: ρ1
2: Codeword Masking using CSPRNG:
a) CSPRNG -> u(1), u(2), ... u(2n)
b) r(i) = x(i)+u(i) (mod 1)
r(1)r(2) r(3)r(4) ... r(2n−1)r(2n)
3: Extreme−Value sequence generation:
n(i) = F−1(r(i))
F(x): CDF of an Log−Weibull random variable
4: Fractional ARIMA sequence Generation:
FARIMA -> s(1), s(2), ..., s(2n)
n(1)n(2)n(3)n(4) ...
s(1)s(2)s(3)s(4)...
5: Combine FARIMA and Extreme−Value Sequences:
a) v(i) = n(i) + s(i)
b) l(i) = g(v(i)), g(v)=b0+b1*v+b2*v^2
l(1)l(2)l(3)l(4) ...
6: Inter−Transmission Time Generation:
T(i) = 2l(i)
T(1)T(2) T(3)T(4) ... T(2n−1)T(2n)

(a)



Message: c(1) c(2) c(3) ... c(n)
6: Reverse Codeword Look Up:
(x(2i−1), x(2i)) −−> c(i)
x(1)x(2) x(3)x(4) ... x(2n−1)x(2n)
Input Parameters:
Weibull Parameters: α, λ
Hurst Parameters: H
Autocorrelation at lag one: ρ1
5: Get codewords by unmasking:
a) CSPRNG -> u(1), u(2), ... u(2n)
b) x(i) = u(i) − r(i) (mod 1)
r(1)r(2) r(3)r(4) ... r(2n−1)r(2n)
4: Get a Uniform (0,1) Random Sequence:
r(i) = F(n(i))
F(x): CDF of an Log−Weibull random variable
n(1)n(2)n(3)n(4) ...
3: Get the Extreme−Value Sequence:
a) v(i) = g−1(l(i))
b) n(i) = v(i) − s(i)
l(1)l(2)l(3)l(4) ...
s(1)s(2)s(3)s(4)...
1: Convert R(i) to log scale:
l(i) = log2(R(i))
2: Fractional ARIMA sequence Generation:
FARIMA -> s(1), s(2), ..., s(2n)
Inter Reception Times at Receiver: R(1)R(2) R(3)R(4) ... R(2n−1)R(2n)

(a)

Figure 5. Encoder

Figure 6. Decoder

The encoder of our HTTP timing channel is detailed in Figure 5(a). A single 8-bit ASCII character c_i will be mapped to two inter-arrival times T_{2i-1}, T_{2i} by this encoder. A message, consisting of a sequence of 8-bit ASCII characters $c_1, c_2, \cdots, c_n$, is encoded in a sequence of TCP new connection inter-arrival times $T_1, T_2, \cdots, T_{2n}$ which have the same marginal distribution and autocorrelations as a legitimate HTTP traffic trace.

The covert message is implanted in the *i.i.d.* Log-Weibull random sequence n_i. The sender and receiver share a code book, a one-to-one mapping of 8-bit binary strings to two-dimensional vectors $(\frac{k_1}{16}, \frac{k_2}{16})$, where k_1 and k_2 are integers between 0 and 15.

The first step of our scheme is to look up the codeword for each character in the message. We use (x_{2k-1}, x_{2k}) to denote the codeword for character c_k. At the end of the first step, the message $\mathbf{msg}$ is transformed to a sequence of numbers $\mathbf{x} = \{x_1, x_2, \cdots, x_{2n-1}, x_{2n}\}$.

In the second step, we use a Cryptographic Secure Pseudo Random Number Generator (CSPRNG) to generate a sequence of pseudo uniform (0,1) random numbers $\mathbf{u} = u_1, u_2, \cdots, u_{2n-1}, u_{2n}$. The seed used by CSPRNG is shared between the sender and receiver, but not with the detector of the covert timing traffic. We then *mask* the sequence $\mathbf{x}$ with $\mathbf{u}$ to obtain a new sequence of numbers $\mathbf{r} = r_1, r_2, \cdots, r_{2n-1}, r_{2n}$ by setting

$$r_k = x_k \oplus u_k \triangleq (x_k + u_k) \bmod 1.$$

In the third step, we create an *i.i.d.* Log-Weibull random sequence $\{n_k, k = 1, 2\cdots\}$ by setting $n_k = F^{-1}(r_k)$, where $F(x)$ is the *c.d.f.* of a Log-Weibull random variable with mean μ_l and variance σ_n^2. This step accomplishes the goal specified in step 12 of our Algorithm 3.1 (in Section 3), that is to generate an *i.i.d.* Log-Weibull random sequence $\{n_j\}$. Additionally, the third step in our design also embeds the covert information in $\{n_j\}$. This sequence $\{n_i\}$ will then be added to a fractional ARIMA sequence $\{s_1, s_2, \cdots\}$ generated in step 4. This fractional ARIMA sequence has the same Hurst parameter as the trace data.

In the last two steps, the fractional ARIMA and the *i.i.d.* Log-Weibull sequence are joined together, and transformed to the inter-arrival time $T_1, T_2, \cdots, T_{2n}$ according to our model introduced in Section 3. The sender then initiates new HTTP connection times according to the values of $T_1, T_2, \cdots, T_{2n}$.

The sender and the receiver share the following using an auxiliary channel prior to initiating the covert communication:

- Code Book: it contains the mapping from 8-bit characters to two-dimensional vectors $(x_1, x_2) = (k_1/16, k_2/16)$, where k_i are integers between 0 and 15, inclusive.

- Traffic Model Parameters α, λ, H, and ρ_1: The underlying assumption is that the sender does the determination of model parameters for legitimate traffic that is similar in characteristic to the legitimate traffic in which he will embed the covert information. For example, the daytime traffic over different weekdays may be statistically similar.

- Seed for CSPRNG: it is used to generate a common CSPRNG sequence.

- Seed for FARIMA series: it is used to generate a common FARIMA sequence.

The procedure for recovering the message at the receiver is simply the reverse of the sender scheme, as illustrated in Figure 6(a). After the receiver get inter-arrival times R_i, it will execute the following tasks:

- 1. Convert R_i to log scale: $l_i = \log_2(R_i)$

- 2. Generate s_i from the FARIMA model using the same seed as the sender, so that the resulting series is identical to that used by the sender.

- 3 a) Obtain the intermediate sequence v_i: $v_i = g^{-1}(l_i)$

  3 b) Obtain the Log-Weibull sequence n_i: $n_i = v_i - s_i$
- 4. Transform the Log-Weibull sequence n_i to a random sequence r_i: $r_i = F(n_i)$, where $F(\cdot)$ is the *cdf* of Log-Weibull distribution
- 5 a). Generate u_i from the CSPRNG using the same seed as the sender

  5 b). Obtain the codeword x_i: $x_i = (r_i - u_i)$ mod 1
- 6. Get character c_k from x_i, x_{i+1} using codebook

At the receiver, the inter-arrival times observed are $R_1, R_2, \cdots, R_{2n}$. These are a distorted version of sender's inter-arrival times $T_1, T_2, \cdots, T_{2n}$. $R_i = T_i + \epsilon_i$, where ϵ_i's are network jitters.

Our initial experiments show most decoding errors occur when the inter-arrival times are small. We conducted more rigorous error analysis and proved that the smaller the inter-arrival time, the less jitter it can tolerate. The detailed error analysis can be found in [6]. Through our experiments and error analysis, we found when the inter arrival times is greater than 100 ms, jitters has much less impact on decoding errors.

Based on this observation, our method of reducing the decoding error is that if a codeword x^* is encoded with a small inter-arrival time, we will re-encode the same character x^* using the next values of CSPRNG and FARIMA sequence until the inter-arrival time T obtained is larger than 100 ms. Note that, using our encoder, the same codeword can be mapped to different inter-arrival times. We will transmit all the inter-arrival times obtained through the encoding process illustrated in Figure 5(a), including the small inter-arrival times that are less than 100 ms so that the desired statistical properties of the traffic is not disturbed. The receiver will record all the inter-reception times, but the decoder will discard the small inter-reception times when recovering the covert information.

In our encoding scheme, it is important that an error in one character does not ripple over to subsequent characters and is contained. In our basic timing channel design, each character is represented by two inter-arrival times. Due to ”re-encoding” in our error correction, a character could be mapped to three or more inter-arrival times. In order to contain the character decoding error, we require even number of inter-arrival times (including small inter-arrival times) to represent one character. This design guarantees that even if the decoder mistakenly discards a ”small” inter-reception time or accepted a ”larger” inter-reception time, it can always start at the right positions to decode characters.

5. Experiments

We implemented this covert timing channel design in Java, using a client/server architecture. The sender injects the covert information into the *i.i.d.* Log-Weibull series, and obtains the desired inter-arrival times T_i according to our design in Figure 5(a). It controls the inter-transmission time by using $Thread.sleep(T_i)$ (T_i is in milliseconds). The accuracy of the $Thread.sleep(T)$ method is 1 ms. The receiver passively collects the TCP packet reception times and decodes the message by extracting the covert information from the inter-reception times R_i, according to the design in Figure 6(a). There is no feedback from receiver to sender regarding when the packet is received or whether it is decoded correctly.

Table 2. Parameter Values for Two SubNets

SubNet 1 Parameters	data 1	data 2	data 3	data 4
α	1.50	1.34	1.20	1.36
λ	0.77	0.74	0.73	0.76
H	0.61	0.52	0.81	0.65
ρ_1	0.12	0.14	0.23	0.18
r (c/s)	0.57	0.62	0.70	0.64
SubNet 2 Parameters	data 1	data 2	data 3	data 4
α	0.51	0.43	0.44	0.45
λ	0.90	0.85	0.87	0.91
H	0.57	0.56	0.59	0.70
ρ_1	0.08	0.10	0.12	0.09
r (c/s)	1.87	2.15	2.12	2.11

We conducted our experiments on two pairs of computers using the PlanetLab environment. The receivers are hosts at Purdue University, and the senders are PlanetLab nodes located at Princeton University and Stanford University. The average RTT between Purdue and Princeton is approximately 39.5 ms, and the average RTT between Purdue and Stanford is approximately 73.5 ms. The average RTT times for both pairs remained stable during the course of ten hour experiments through the day.

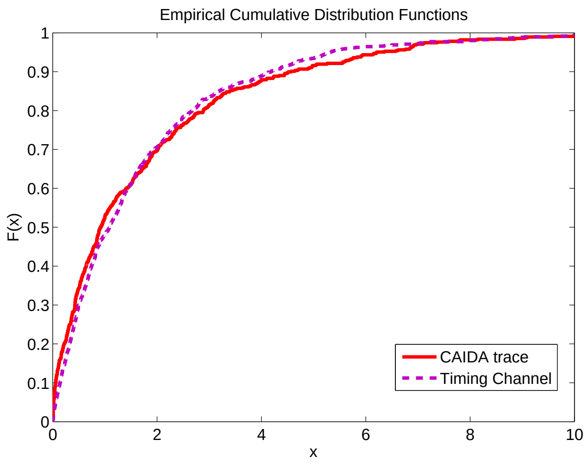
In our experiments, we used parameters estimated from two subnets. Each subnet has four data sets, each containing about 15-minutes traffic trace. The data was collected by CAIDA in March 2009. Further details of the data set have been presented in Section 3. We will verify through our experiments if our covert timing channel traffic is statistically indistinguishable from these real data, and if it can avoid detection.

The values of input parameters for the eight data sets, α, λ, H and ρ_1, all estimated directly from these trace data, are listed in Table 2. We create 8 timing channels corresponding to the 8 traces. The sender sends a text file of 410 characters over the covert channels mimicking subnet 1, and a text file of 1100 characters over the covert channel mimicking subnet 2. We will see shortly that the timing channels mimicking subnet 2 has a higher data rate than subnet 1. We ran a set of the eight timing channels from Princeton to Purdue. The inter-arrival times from each of these eight timing channels are later used to test if these timing channels can avoid the best available detection schemes, such as the entropy based detection and the Kolmogorov-Smirnov Test.

Additionally, we ran two timing channels hourly over the course of a day on two pairs of hosts to see how network conditions impact the decoding error. The first timing channel runs from Stanford University to Purdue University, mimicking the first subnet. The second timing channel runs from Princeton University to Purdue University, mimicking the second subnet. We found the decoding error ranges from 2.9% to 6%. The data rate for the covert channels mimicking the first subnet's traffic is approximately 2 bits/sec; the data rate for the second covert channel is approximately 6 bits/sec.

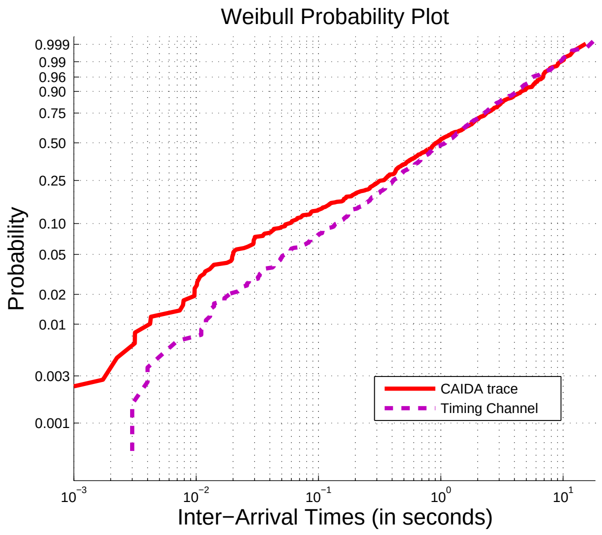
The data rate for our covert timing channel is largely determined by $E[t_j]$, the mean value of the consecutive new HTTP connection request times. If one character (8-bits) is encoded in two inter-arrival times, the data rate of our timing channel is approximately 8/(2$E[t_j]$) = 4r b/s. Because of the use of error-correction in our timing channel, small inter-arrival times do not carry information, and one character could be mapped to $2N$ inter-arrival times, where N is an integer and $N \geq 1$. This reduces the actual data rate to be less than the maximum achievable rate of $4r$ b/s. As shown in Table 2, $r = 2.12c/s$ for data 3 in subnet 2. The data rate of this timing channel is 6 bits/sec, less than $4r$ b/s. Although the data rates are not very high, we would caution the reader not to underestimate the potential security risks. Since these timing channels mimic non-stationary LRD HTTP traffic, it can potentially be used long-term without detection. Further, often systems have small-sized private data items.

We compare the traffic trace from our timing channels and the real data. Figures 7(a) and 9(a) compare the first order and the second order statistics of the data trace from our timing channel with its corresponding real data trace (data 1 of subnet 1 is used[5]) . Figure 7(a) shows that the empirical distributions of the two traces are very close to each other. In fact, the maximum distance between these two empirical distributions using the Kolmogrov-Smirnov test is only 0.066. Figure 9(a) shows the PSD estimates of the covert traffic and the legitimate traffic, and they are very close to each other. The closeness of the PSD indicates the closeness of their autocorrelations functions.



(a)

Figure 7. Empirical *cdf* of the inter-arrival times from our covert channel and CAIDA data (Subnet 1, data 1)

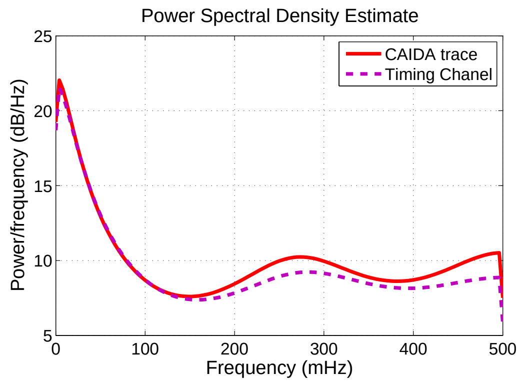


(a)

Figure 8. Weibull probability plots

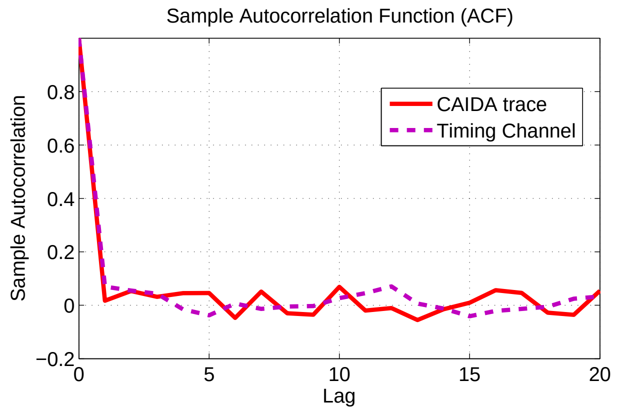
Next, we evaluate how well our timing channel can evade current detection methods. First, we conduct two sample Kolmogorov-Smirnov tests (KS test) on each covert traffic and real data pair. The two sample KS test uses the maximum distance between two empirical distributions, $KS\_STAT = \max(|F1(x) - F2(x)|)$, where $F1(x)$ and $F2(x)$ are empirical distributions of the real data and the covert channel data. Table 3 shows the values of KS_STAT and corresponding p-values for each pair. When the allowable false alarm level is set to 1% as in [3], none of our timing channel traffic can be detected since all the p-values are greater than 1%.

[5]Data 4 of subnet 1 was used in Section 3.



(a)

Figure 9. PSD Estimates of the log inter-arrival times from our timing channel and CAIDA data (Subnet 1, data 1)



(a)

Figure 10. Sample Autocorrelation Functions

Table 3. Kolmogorov-Smirnov Test

SubNet 1	data 1	data 2	data 3	data 4
KS-STAT	0.0657	0.0845	0.0637	0.0516
p-value	0.096	0.012	0.0863	0.2678
detect	no	no	no	no
SubNet 2	data 1	data 2	data 3	data 4
KS-STAT	0.0402	0.0442	0.0466	0.0513
p-value	0.2088	0.1088	0.08	0.043
detect	no	no	no	no

The regularity test proposed in [1] checks if the variance of the inter-arrival times is relatively constant, and the traffic is flagged as a covert timing channel if the variance remains constant. Our timing channel is designed to mimic the non-stationary LRD HTTP traffic. More specifically, our timing channel traffic patterns match the legitimate HTTP traffic patterns in each small time interval (say 15 minutes). The marginal distributions of our timing channels match that of the changing legitimate HTTP traffic patterns. Therefore, the variance of the inter-arrival times of our timing channels change the same way as the legitimate HTTP traffic patterns change. Thus, by design, our timing channel can evade the regularity test independent of the threshold used in the test.

A recent detection scheme based on entropy is proposed in [3], where data is first partitioned into Q equiprobable bins according to a desired cumulative distribution function $F(x)$. The bin numbers y for the data value x can be calculated using $y = \lfloor F(x) * Q \rfloor$, and then used for the entropy and the corrected conditional entropy computation. By using a single *cdf* for legitimate traffic, they implicitly assume that the legitimate traffic is stationary. The *c.d.f.* of a non-stationary traffic can change over time, so their detection method does not apply directly to timing channels mimicking non-stationary traffic.

Their recommended value for fine binning $Q = 65536$ also fails to work for small data sets with

Table 4. Entropy Values for Covert Traffic 1

traffic type	data 1	data 2	data 3	data 4
training data	6.81	6.80	6.85	6.82
legit	6.56	6.50	6.50	6.51
covert 1	6.85	6.84	6.87	6.89
p-value	0.33	0.36	0.28	0.24
detect?	no	no	no	no

Table 5. Entropy Values for Covert Traffic 2

traffic type	data 1	data 2	data 3	data 4
training data	6.93	6.94	6.95	6.95
legit	6.29	6.38	6.44	6.34
covert 2	6.86	6.88	6.90	6.87
p-value	0.55	0.52	0.48	0.53
detect?	no	no	no	no

less than 65536 data points. The reason is that legitimate traffic is expected to be uniformly distributed among the Q bins, thus having entropy value near 16 when $Q = 65536$. Traffic with entropy values less than 15.12 in [3] is flagged as covert traffic. Consider legitimate traffic that follows exactly the desired *c.d.f*, but only has 2000 inter-arrival times. Since the maximum entropy for any data sets with 2000 data points is $\log_2(2000) = 10.97$, the entropy for this legitimate traffic is far less than the desired entropy 16, and will be falsely flagged as covert traffic.

In each of our 15-minute data sets from subnet 1, there are approximately 600 HTTP connections. It is appropriate to use $Q = 128$ for fine binning. We use $Q = 5$ to calculate CCE values as in [3]. Table 4 and 5 show the entropy values of covert traffic mimicking subnet 1 and subnet 2 respectively, compared with the legitimate traffic and the training data. Table 6 and 7 show the CCE values of covert traffic mimicking subnet 1 and subnet 2 respectively, compared with legitimate traffic and training data. The training data are the CAIDA data set that was used to obtain the model parameters for the covert timing channels. The p-values are calculated for each entropy or CCE value using T-test. The T-test was applied to determine if the traffic coming from the covert channel differs significantly from the ”normal” traffic, where normal traffic included the training and the legitimate traffic traces. As shown in these tables, all the 8 covert timing channels evade entropy and CCE tests, even if the allowable false alarm level is 5%.

Table 6. CCE Values for Covert Traffic 1

traffic type	data 1	data 2	data 3	data 4
training data	2.21	2.19	2.17	2.21
legit	1.84	1.81	1.82	1.84
covert 1	2.22	2.23	2.18	2.21
p-value	0.35	0.33	0.44	0.37
detect?	no	no	no	no

Table 7. CCE Values for Covert Traffic 2

traffic type	data 1	data 2	data 3	data 4
training data	2.26	2.21	2.22	2.22
legit	1.75	1.85	1.88	1.80
covert 2	2.25	2.25	2.24	2.22
p-value	0.37	0.37	0.39	0.43
detect?	no	no	no	no

6. Conclusion

Internet traffic has often been show to display LRD characteristics. Thus, traditional covert channel schemes can easily be detected by comparing their traffic characteristics with Internet traffics. T o overcome this problem, we have designed a covert timing channel scheme that can mimic legitimate traffics displaying LRD property. We show that our covert timing channel can be hidden in the Web traffic, the most observed traffic on Internet today. We used the HTTP new connection inter-arrival times to carry the covert information. We found that the marginal distribution and autocorrelation functions of the inter-arrival times from our covert timing channel matched closely with that from recent traces of real traffic.

We implemented our design and have conducted extensive experiments on the PlanetLab nodes and verified the close match of our covert traffic with the real the data. Further, our experiments show that our our covert timing channels evade the current best available detection methods. The data rates of our covert channels range from 2 to 6 bits/sec, and decoding errors range from 3% to 6%.

There are several interesting future directions for this work. One is to develop timing channels for short range dependent (SRD) traffic; and the other is to design a covert timing channel to mimic other commonly used traffics, such as peer-to-peer traffic.

References

[1] S. Cabuk, C. E. Brodley, and C. Shields "IP covert timing channels: design and detection," *Proceedings of 11th ACM conf. Computer and communication security*, pp. 178 – 187, New York, 2004

[2] V. Berk, A. Giani, and G. Cybenko "Detection of covert channel encoding in network packet delays," *Technical Report, TR2005-536, Dartmouth College, 2005*

[3] S. Gianvecchio and H. Wang "Detecting Covert Timing Channels: An Entropy-Based Approach," *Proceedings of 14th ACM conf. Computer and communication security*, 2007

[4] G. Shah, A. Molina and M. Blaze "Keyboard and covert channels," *USENIX Security Symposium*, 2006

[5] S. H. Sellke, C. C. Wang, S. Bagchi, and N. B. Shroff, "Covert TCP/IP Timing Channels: Theory to Implementation," *IEEE INFOCOM*, 2009

[6] S. H. Sellke, C. C. Wang, S. Bagchi, and N. B. Shroff, "Camouflage Timing Channels in Web Traffic," Purdue University ECE Technical Report, 2009, available at http://docs.lib.purdue.edu/ecetr/

[7] W. M. Hu “Reducing Timing Channels with Fuzzy Time,” *in Proceedings of the IEEE Symposium in Security and Privacy*, May. 1991

[8] M. H. Kang, I. S. Moskowitz, and D. C. Lee “A network version of the pump,” *in Proceedings of the IEEE Symposium in Security and Privacy*, May. 1995

[9] J. Giles and B. Hajek “An Information-theoretic and game-theoretic study of timing channels,” *IEEE Trans. on Inform. Theory,* VOL. 48, Sep. 2002

[10] W. S. Cleveland, D. Lin, and D. X. Sun, “IP Packet Generation: Statistical Models for TCP Start Times Based on Connection-Rate Superposition,” *in Proceedings of ACM SIGMETRICS, 2000*

[11] J. Cao, W. S. Clevland, D. Lin, and D .X .Sun “On the Nonstationarity of Internet Traffic,” *in Proceedings of ACM SIGMETRICS, 2001*

[12] K. Park and W. Willinger, *Self-Similar Network Traffic: An Overview,* Book Chapter in Self-Similar Network Traffic and Performance Evaluation, John Wiley & Sons, Inc, 2002

[13] W. E. Leland, M. S. Taqqu, W. Willinger, and D. Wilson, *On the Self-Similar Nature of Ethernet Traffic,* in IEEE Transactions on Networking, Feb 1994

[14] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson “Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level,” *in IEEE/ACM Transactions on Networking,* 1997

[15] J. R. M. Hosking, *Fractional Differencing,* in Biometrika, 1981

[16] V. Paxson and S. Floyd “Wide-area traffic: the failure of Poisson modeling,” *IEEE tran. Networking,* May 1991

[17] V. Paxson, “Fast, Approximate Synthesis of Fractional Gaussian Noise for Generating Self-Similar Network Traffic, ” *Computer Communication Review 27(5),* pp. 5-8, Oct. 1997.

[18] T. Karagiannis, M. Molle, and M. Faloutsos, “Long-Range Dependence: Ten Years of Internet Traffic Modeling, ” *in IEEE Internet Computing, 2004*

[19] T. Karagiannis and M. Faloutsos, *SELFIS: A Tool for Self-Similarity and Long-Range Dependence Analysis*, in 1st workshop on Fractals and Self-Similarity in Data Mining: Issues and Approaches (in KDD), July 23, 2002

[20] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Petterson, M. Wawrzoniak, and M. Bowman “Planetlab: an overlay testbed for broad-coverage services,” *in SIGCOMM Comput. Commun. Rev. 33,3,* (2003), 3-12

[21] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, F. Jahanian, and M. Karir “ATLAS Internet Observatory 2009 Annual Report,” *available at http://www.nanog.org/meetings/nanog47/presentations/Monday/Labovitz_ObserveReport_N47_Mon.pdf*

[22] Colby Walsworth, Emile Aben, kc claffy, Dan Andersen, The CAIDA Anonymized 2009 Internet Traces DataSet, *available at http://www.caida.org/data/passive/passive_2009_dataset.xml*

A. Appendix: Decoding Errors

In Section 4, we presented our design of a covert timing channel that can mimic LRD traffic, based on a model we proposed in Section 3. The design of our timing channel encoder that converts a character c_k to two inter-arrival times T_{2k-1}, T_{2k} (for $k = 1, 2, \cdots, n$), is illustrated in Figure 5(a). Due to network jitters, the inter-reception times R_{2k-1}, R_{2k} obtained by the receiver are slightly distorted from T_{2k-1}, T_{2k}, and $R_j = T_j + \epsilon_j$ (ϵ_j are network jitters).

The decoder in Figure 6(a) uses $R_1, R_2, \cdots, R_{2n}$ to recover the covert message. Our initial experiments show most decoding errors occur when the inter-arrival times T_j are small. Here, we present a more rigorous error analysis and show that the smaller the inter-arrival time, the less jitter it can tolerate.

Let $l_j^D = \log_2(R_j) = log_2(T_j + \epsilon_j)$, and recall that $l_j = \log_2(T_j)$. We have

$$\Delta l_j = l_j^D - l_j = \log_2(T_j + \epsilon_j) - \log_2(T_j) \tag{5}$$

By the mean value theorem, there is a $T^* \in (T_j, T_j + \epsilon_j)$ such that

$$\log_2(T_j + \epsilon_j) - \log_2(T_j) = \frac{\epsilon_j}{\ln(2)T^*} \tag{6}$$

By Equations (5) and (6), we have

$$\Delta l_j = \frac{\epsilon_j}{\ln(2)T^*} \tag{7}$$

Recall that $\{n_j\}$ is an *i.i.d.* Log-Weibull random sequence, and $\{s_j\}$ is a a LRD FARIMA sequence. In our encoder, we used an intermediate random variable $v_j = n_j + s_j$ to obtain the Log-Weibull distributed l_j, using $l_j = b_0 + b_1 \cdot v_j + b_2 \cdot v_j^2$. Let v_j^D satisfies $l_j^D = b_0 + b_1 \cdot v_j^D + b_2 \cdot (v_j^D)^2$. and $\Delta v_j = v_j^D - v_j$. Then, we have $\Delta l_j = b_1 \Delta v_j + b_2 \cdot (v_j + v_j^D) \cdot \Delta v_j$

Based on the values of b_1 and b_2, calculated according to [10], $\Delta l_j \approx \Delta v_j$.

Dente $n_j^D = v_j^D - s_j$ in step 3b of the decoder in Figure 6(a). Recall that $\{s_j\}$ is shared between the encoder and decoder since they share the seed for generating the sequence, and $v_j = n_j + s_j$. We have,

$$\Delta l_j \approx \Delta n_j = n_j^D - n_j \tag{8}$$

In step 3 of Figure 5(a), $n_j = F^{-1}(r_j)$, where $F^{-1}(x)$ is the inverse function of *c.d.f.* of the Log-Weibull distribution. Therefore,

$$\Delta n_j = -b \ln(-\ln(r_j^D)) + b \ln(-\ln(r_j)), \tag{9}$$

where $b = -1/(\ln(2) \cdot \lambda)$, and λ is the shape parameter of the Weibull distribution.

Note $r_j = u_j + x_j$ (mod 1) $= u_j \oplus x_j$ from step 2b of Figure 5(a). By equations (7), (8), and (9), we have

$$\frac{-\epsilon}{ln(2) \cdot b \cdot T_j} = \ln(-\ln(u_j \oplus x_j^D)) - \ln(-\ln(u_j \oplus x_j)) \tag{10}$$

Thus,

$$\frac{-\epsilon}{ln(2) \cdot b \cdot T_j} = \ln \frac{\ln(u_j \oplus x_j^D)}{\ln(u_j \oplus x_j)} \tag{11}$$

So that,

$$\frac{\ln(u_j \oplus x_j^D)}{\ln(u_j \oplus x_j)} = \exp\{\frac{-\epsilon}{ln(2) \cdot b \cdot T_j}\} \tag{12}$$

Denote

$$\beta = \exp\{\frac{\epsilon_j}{\ln(2) \cdot b \cdot T_j}\} \tag{13}$$

We have

$$u_j \oplus x_j^D = ((u_j \oplus x_j)^\beta = r_j^\beta$$

$$\Delta x_j = x_j^D - x_j = (x_j^D \oplus u_j) - (x_j \oplus u_j) = r_j^\beta - r_j,$$

In order for our decode to decode correctly, we would like to have $|\Delta x_j| < 1/32$. When $0.92 < \beta < 1.08$, we have $|\Delta x_j| < 1/32$ regardless of the values of u_j.

By equation (13), we have

$$|\frac{\epsilon_j}{\ln(2) \cdot b \cdot T_j}| = |\ln(\beta)| \tag{14}$$

Thus,

$$|T_j| = |\frac{\epsilon_j}{\ln(2) \cdot b \cdot \ln(\beta)}| \tag{15}$$

Since $b = -1/\ln(2) \cdot \lambda$, we have

$$|T_j| = |\frac{\lambda \epsilon_j}{\ln(\beta)}| \tag{16}$$

We can see from Equation (16) that if T_j is too small, $\ln(\beta)$ will be too large, causing β to be outside the interval (0.92, 1.08) for correct decoding. Using the Weibull parameter $\lambda = 0.9$ in our data 1 from Subnet 2 (Table 2, Section 5), and choosing $\beta = 1.08,$ or 0.92, the worst case scenario, we have $1/\ln(\beta) \approx 12$, and Equation(16) gives

$$|T_j| > (12\lambda)\epsilon_j > 10\epsilon_j \tag{17}$$

If the maximum jitter in the network is 10 ms, We will not have decoding error when $T_j > 100$ ms. Our analysis show that small T_j has less tolerance for network jitters, and can cause decoding errors. Our proposed solution is to add redundancy and if a character is encoded with a small T value, we will re-encode this character until it is encoded with a larger T value.