Stealthier Inter-packet Timing Covert Channels

Sebastian Zander, Grenville Armitage, Philip Branch

Centre for Advanced Internet Architectures (CAIA)
Swinburne University of Technology
Melbourne Australia
{szander, garmitage, pbranch}@swin.edu.au

Abstract. Covert channels aim to hide the existence of communication. Recently proposed packet-timing channels encode covert data in inter-packet times, based on models of inter-packet times of normal traffic. These channels are detectable if normal inter-packet times are not independent identically-distributed, which we demonstrate is the case for several network applications. We show that ~80% of channels are detected with a false positive rate of 0.5%. We then propose an improved channel that is much harder to detect. Only ~9% of our new channels are detected at a false positive rate of 0.5%. Our new channel uses packet content for synchronisation and works with UDP and TCP traffic. The channel capacity reaches over hundred bits per second depending on overt traffic and network jitter.

Keywords: Covert Channels, Steganography, Inter-packet Times

1 Introduction

The very fact that two parties are communicating can be actionable information, even if an external observer cannot determine the contents of the messages being passed. Covert channels aim to hide the very existence of communication, enabling parties to convey information undetected [1]. Individuals and groups may have various reasons to utilise covert channels (e.g. for communication or bypassing access controls), often motivated by the existence of adversarial relationships. Examples include government agencies versus criminal organisations, hackers versus company IT departments, or dissenting citizens versus their governments.

The prisoner problem is the de-facto model for covert channel communication [1]. *Alice* and *Bob* are thrown into prison and intend to escape. To agree on an escape plan they need to communicate, but a *warden* monitors all their messages. If the warden finds any signs of suspicious messages she will place Alice and Bob into solitary confinement – making it impossible for them to escape. Alice and Bob must exchange innocuous messages containing hidden information that hopefully the warden will not notice.

Recently researchers proposed covert channels encoded in inter-packet times, also referred to as Inter Packets Gaps (IPGs). Gianvecchio *et al*. [2] and Sellke *et al*. [3] proposed channels that are hard to detect because they perfectly mimic the shape of IPG distributions of real applications. However, they are hard to detect only if IPGs are independent identically-distributed (iid). Paxson *et al*. showed that Telnet traffic exhibits this behaviour [4]. However, we found that IPGs of several applications, such

as UDP-based game and VoIP traffic and TCP traffic, often are not iid because there is auto-correlation. The channel proposed in [2,3] is easy to detect with such applications.

The channel in [2,3] requires accessible sequence numbers in the overt traffic. Otherwise any lost packets desynchronise covert sender and receiver. TCP provides sequence numbers, but not all UDP-based traffic has sequence numbers, or they may not be accessible if the protocol is encrypted.

We propose an improved channel, based on techniques for information hiding in images (steganography), which is harder to detect when IPGs are not iid. Our new technique generates the random numbers needed for encoding from the packets themselves, which makes the channel robust enough for use with all UDP-based protocols.

First, we motivate our work by demonstrating that several applications have auto-correlated IPGs. We then present the improved covert channel. We show that for traffic with auto-correlated IPGs the existing timing channel is easy to detect with ~80% accuracy and a false positive rate of 0.5%[1]. Our new channel is much harder to detect. The detection accuracy reduces to only ~9% with a false positive rate of 0.5%. A drawback of our new channel is a reduced robustness against network jitter. However, based on a proof-of-concept implementation we show that the channel capacity is still high enough for practical use, even across uncongested Internet paths with more than 10 hops. The capacity ranges from a few bits per second to over hundred bits per second, depending on the overt traffic's packet rate and network jitter.

The paper is organised as follows. Section 2 outlines related work. In Section 3 we show that several applications often have auto-correlated IPGs. In Section 4 we propose our new channel. In Section 5 we analyse the detection accuracy for the channel in [2,3] and our improved channel using machine learning. In Section 6 we analyse the capacity of our improved channel. Section 7 concludes and outlines future work.

2 Related Work

The possibility of encoding covert channels in the timing of packets (or frames) was identified early by Padlipsky *et al*. [5]. However, all the timing channels proposed prior to Berk's work [6] were based on encoding in varying packet rates over time as opposed to encoding in IPG values directly. For space reasons we do not discuss them here and instead refer the reader to [1].

Berk *et al*. introduced packet-timing channels where the covert information is encoded in the IPGs of consecutive packets [6]. They compared channels with two IPGs and multiple IPGs, and developed a mechanism by which the sender can pick the optimal symbol distribution in multi-symbol channels. Sha *et al*. developed a "bug" that hooks into the connection between keyboard and computer and ex-filtrates all keystrokes by modulating the IPGs of network traffic send by the victim [7]. Gianvecchio *et al*. later showed that both of these channels are easy to detect [8].

Gianvecchio *et al*. [2] developed a stealthier IPG timing channel and evaluated its performance. They proposed to fit a model to the IPG distribution of real traffic and then use the model to generate a covert channel with identical distribution. If the IPGs

[1] In reality most flows are normal flows, so a detector's false positive rate must be low or covert channels are effectively masked (see Section 5).

of normal traffic are iid this channel cannot be detected. Sellke *et al*. proposed another scheme for encoding covert data in IPGs and evaluated its performance based on experiments across the Internet [3]. Similar to [2] they also showed that timing channels can be made indistinguishable from normal traffic by mimicking normal iid IPGs.

Luo *et al*. proposed to encode covert data into the length of TCP data bursts, where a data burst is a number of TCP segments sent before the next TCP ACK arrives [9]. The channel is robust against packet jitter, loss and reordering, but has very low capacity. Furthermore, its stealth is low, as flows with covert channel behave very different from normal flows [9]. Liu *et al*. introduced a channel that encodes covert data such that the normal distribution of IPGs is closely approximated and spreading techniques are used to provide robustness [10]. Their covert channel is hard to detect with simple shape and regularity tests, however these tests are known to be insufficient [8].

3 Inter-packet Times Analysis

We analyse the IPGs of several applications. Our results illustrate the existence of IPG auto-correlation for certain types of traffic.

3.1 Datasets and methodology

We analysed two UDP-based applications, the First Person Shooter (FPS) game Quake III Arena (Q3) [11] and the VoIP application Skype. The Q3 client-to-server traffic was taken from trace files collected at our public game server. It contains traffic from 106 clients. The Q3 server-to-client traffic analysis is based on trace files captured at a client that connected to 224 different game servers. We analysed the IPGs of 44 traffic flows of Skype one-to-one calls collected by Branch *et al*. [12].

We also analysed a mix of UDP and TCP traffic from the Twente traces [13]. Since the trace does not contain payload data we do not know the applications. Based on the port numbers it seems most of the TCP flows were bulk-transfers: HTTP, FTP, or peer-to-peer file sharing applications, such as BitTorrent. A large part of the UDP flows were game traffic, for example Half-Life/Counterstrike and Quake. In total the dataset has 111 UDP and 214 TCP unidirectional traffic flows from different sources.

For each flow we computed the auto-correlation over the first 5 000 IPGs. Auto-correlation is the cross-correlation of a time series with itself. Let x_t be the IPG value at time t, let μ and σ be the mean and variance over all t, let τ be the time lag and $\mathrm{E}(.)$ be the expectation value. The auto-correlation function (ACF) for one traffic flow is:

$$R(\tau) = \frac{\mathrm{E}\left((x_t - \mu)(x_{t+\tau} - \mu)\right)}{\sigma^2} \ . \tag{1}$$

For a set of n traffic flows of one dataset we compute the ensemble average over all ACFs (referred to as *average ACF*). The average ACF at lag τ is defined as the mean of the absolute ACF values of each flow at τ:

$$\bar{R}(\tau) = \frac{1}{n} \sum_n |R_i(\tau)| \ . \tag{2}$$

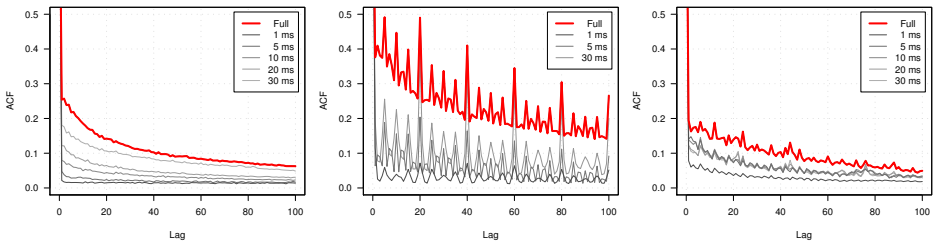

Fig. 1. Average auto-correlation for Q3 client-to-server (left), Skype (middle) and TCP (right) traffic (zoomed y-axis)

We analyse the auto-correlation of IPGs and least significant parts of IPGs. We define the least significant part as:

$$d_{\text{lsp}} = d \bmod l = d - \left\lfloor \frac{d}{l} \right\rfloor l \,, \tag{3}$$

where d is the IPG and l is the size of the least significant part. For example, if the IPG is 21.75 ms and $l = 1$ ms then $d_{\text{lsp}} = 0.75$ ms (sub-millisecond part).

3.2 Results

Figure 1 shows the average ACFs of Q3 client-to-server, Skype and TCP traffic for the full IPGs and decreasing least significant parts ($\tau \leq 100$). The average ACF of the full IPGs decays more rapidly than individual ACFs (e.g. the one shown in Figure 2(left)), as individual ACFs have lows and highs at different places. Still it is significantly larger than the average ACF for small least significant parts.

For the UDP-based applications IPGs are often at least moderately correlated. However, smaller least significant parts of IPGs are largely uncorrelated. The size of the least significant part where correlation diminishes depends on the application. The TCP flows show less correlation, but many flows still have low to moderate correlation.

IPGs of applications where packet send times are human-driven and effectively random were shown to be iid (e.g. Telnet [4]). However, for other applications auto-correlation of IPGs tends to exist because of large-time-scale behaviours, such as the congestion window growth/collapse for TCP or the application's encoding for games or VoIP over UDP. Network jitter reduces existing correlations, but even after many hops often there is some correlation. Furthermore, if the warden is close to the covert sender, the IPGs observed are largely unaffected by network jitter.

The small time-scale behaviour (exposed by looking only at the least significant part of IPGs) is jittered by largely uncorrelated noise, for example queuing delays at each hop. Our new encoding technique exploits this effect.

4 Covert Channel

We first review the previous encoding scheme on which our improved scheme is based and then propose our improved technique.

4.1 Basic encoding

In Gianvecchio's and Sellke's encoding scheme [2, 3] the covert sender (*Alice*) and receiver (*Bob*) share a model of the IPGs, which was previously generated based on an analysis of real traffic. The model can be a histogram of measured IPGs or a standard statistical distribution fitted to the observed IPGs.

Alice and Bob also need synchronised random number generators. They have to agree on a Cryptographically Secure Pseudo Random Number Generator (CSPRNG) and a seed value. The seed value can be derived from Alice's and Bob's shared secret.

Let $R = r_1, r_2, \ldots, r_n$ be the sequence of Uniform(0,1) random numbers generated and s be a symbol out of the set of possible symbols S. For example, a binary channel has two symbols: $S = \{s_1, s_2\} = \{0, 1\}$. Then Alice encodes covert bits as follows. Each discrete symbol is transformed into a continuous symbol:

$$F_{\text{cont}}(s, r) = \left(\frac{s}{|S|} + r\right) \bmod 1 = r_{\text{s}} \ . \tag{4}$$

The IPGs $d_1, d_2, \ldots, d_n$ are produced by the encoding function:

$$F_{\text{enc}} = F^{-1}_{\text{model}}(r_{\text{s}}) = d \ , \tag{5}$$

where F^{-1}_{model} is the inverse distribution function of the model. Bob decodes the bits from the observed IPGs $\tilde{d}_1, \tilde{d}_2, \ldots, \tilde{d}_n$, which is Alice's series modified by timing noise, such as timing inaccuracies at the sender, network jitter and measuring inaccuracies at the receiver. Bob decodes the continuous symbol by applying:

$$F_{\text{dec}} = F_{\text{model}}\left(\tilde{d}\right) = \tilde{r}_{\text{s}} \ , \tag{6}$$

where F_{model} is the distribution function of the model. The discrete symbol received is:

$$F_{\text{disc}}(\tilde{r}_{\text{s}}, r) = |S| \cdot ((\tilde{r}_{\text{s}} - r) \bmod 1) = \tilde{s} \ . \tag{7}$$

This encoding scheme generates an IPG distribution that is indistinguishable from the IPG distribution of normal traffic if normal IPGs are iid [2, 3]. However, as we showed in the previous section this assumption is often not true. This makes the channel detectable. Figure 2 shows the ACF of the IPGs of normal Q3 client-to-server traffic on the left and the ACF of the IPGs of the covert channel in the middle. The IPG histograms of normal traffic and covert channel (not shown) look alike, but the difference of the ACFs is striking.

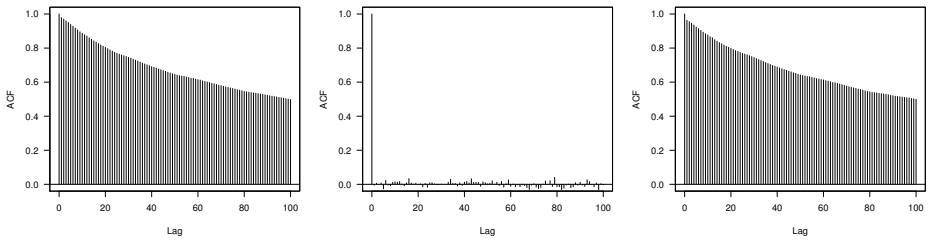

Fig. 2. Auto-correlation of IPGs of normal Q3 traffic (left), covert channel in [2, 3] (middle) and sub-band encoding with $l = 5$ ms (right)

4.2 Improved synchronisation

The basic encoding works only if the overt traffic has accessible sequence numbers. Otherwise any packet loss permanently desynchronises Alice and Bob. Furthermore, if Bob is unable to observe a transmission's start he can never synchronise with Alice.

Synchronisation can be improved by computing R from the packets themselves. Let H be a good hash function that maps the inputs as evenly as possible over the output range (uniform distribution). Let B denote some part of an overt packet that is immutable on the path from Alice to Bob, and let b_i be the value of B for the i-th packet. Random numbers are generated as follows:

$$r_i = H(b_i) \ . \tag{8}$$

We assume that the inputs b_i vary sufficiently so that the output of H is approximately uniformly distributed. Previous work on packet sampling and one-way delay measurement showed that generally this is the case if B is chosen properly, and suggested several suitable choices for H and B [14]. TCP retransmissions may cause the same inputs b_i repeatedly, but using TCP header information Alice and Bob can 'ignore' them.

Alice and Bob need to agree on H and B. It is possible for a warden to guess H and B and therefore to detect the covert channel, since the choices for H and B are limited. Alice and Bob can use keyed hash functions used for Message Authentication Codes (MACs) for higher security.

With our improved synchronisation technique lost packets can still cause lost bits, but never a permanent desynchronisation.

4.3 Sub-band encoding

Now we present our new encoding scheme that is much stealthier if IPGs are not iid. The scheme encodes covert bits only into the least significant part of IPGs as defined in Equation 3.

Let l be the size of the least significant part of the IPGs. The parameter l determines the trade-off between stealth and robustness. Let D be the range of IPGs (maximum minus the minimum). Then an IPG distribution typically spans $m = \lceil D/l \rceil$ *sub-bands* of

Algorithm 1 Sender and receiver algorithm for sub-band encoding

function encode(packet, prev_pkt_time, pkt_time)
  b = hash_input(packet)
  orig_ipg = pkt_time − prev_pkt_time
  base_ipg = $\lfloor \texttt{orig\_ipg}/l \rfloor \cdot l$
  band = BL $[d_{\text{start}} = \texttt{base\_ipg}]$ .x
  bits = get_bits()
  $r = H_{\text{R}}(b)$
  $\texttt{ipg} = F_{\text{enc}}^{(\text{band})}(F_{\text{cont}}(\texttt{bits}, r)) + \texttt{base\_ipg}$
  return ipg

function decode(packet, prev_pkt_time, pkt_time)
  b = hash_input(packet)
  ipg = pkt_time − prev_pkt_time
  base_ipg = $\lfloor \texttt{ipg}/l \rfloor \cdot l$
  lsp = ipg − base_ipg
  band = BL $[d_{\text{start}} = \texttt{base\_ipg}]$ .x
  $r = H_{\text{R}}(b)$
  $\text{bits} = F_{\text{disc}}\left(F_{\text{dec}}^{(\text{band})}(\texttt{lsp}), r\right)$
  return bits

size l, although in the extreme case it may fit in only a single sub-band. The location of the sub-bands, the IPG value marking the start of each band, depends on the start value of the first band. The location must be selected carefully to minimise the error rate for a given IPG distribution.

For each of the sub-bands the basic encoding scheme is used. However, now we have one probability distribution of the least significant part of the IPGs for each sub-band j, resulting in $F_{\text{enc}}^{(j)}(.) = F_{\text{model}}^{-1(j)}(.)$ and $F_{\text{dec}}^{(j)}(.) = F_{\text{model}}^{(j)}(.)$. When building the model from the example traffic a distribution of the least significant part of the IPGs must be estimated for each sub-band. If the example traffic is only a small sample and the number of bands is large there may be only a few or even zero values for some bands. For such sub-bands as well as sub-bands outside the range covered by the example traffic our algorithm augments the data with uniformly distributed random values so that a minimum number of samples is reached.

Algorithm 1 shows the encoding algorithm. Let BL be a set of tuples $\left(d_{\text{start}}^{(j)}, x_j\right)$ that associate each sub-band index x_j with an absolute IPG value $d_{\text{start}}^{(j)}$ marking the start of the band (*base IPG*). First, the algorithm determines the actual sub-band based on the original unmodified IPG. Then the modified IPG is the sum of the least significant part as given by the sub-band model and the base IPG of the sub-band. Algorithm 1 also shows the decoding. The receiver selects the sub-band based on the observed IPG. Then it computes the least significant part of the IPG and decodes the bits as before.

Figure 2(right) illustrates the effectiveness of sub-band encoding. The covert channel only very slightly decreases the auto-correlations. The drawback of sub-band encoding is that it is less robust against network jitter, but network jitter is often small even on longer paths. Our results in Section 6 show that the channel capacity is still high enough for practical use, even across uncongested Internet paths with more than 10 hops. However, the scheme is also less robust against an active warden that can introduce artificial jitter.

For active channels, where the covert sender also generates the overt traffic, sub-band encoding requires models that model the shape of the distribution as well as the auto-correlations, for example autoregressive integrated moving average (ARIMA) models. The sender first uses the model to create a series of IPGs, and then uses the `encode()` function to encode the covert data. For passive channels, where the covert sender encodes the channel into existing traffic of unwitting hosts, the correlations are already present in the intercepted traffic.

5 Detection

We use classifiers constructed by a supervised Machine Learning (ML) algorithm. During training supervised ML techniques build classifiers based on data instances with class labels attached, so that the data instances are 'optimally' separated into the different classes based on characteristics (*features*) of the instances other than the class label. The classifiers are then used to classify data instances of unknown class.

5.1 Datasets and Features

As normal traffic we used the datasets from Section 3. The covert traffic was created based on these datasets using sub-band encoding. For each normal flow we generated one corresponding covert channel based on the same IPG distribution. We used the first 5 000 IPGs of each flow. The covert data was uniformly random distributed, as if Alice and Bob used encryption. Each dataset had the same number of covert and normal flows to avoid bias of the classifier towards a larger class.

Let $X = [X_1, \ldots, X_n]$ be a series of IPGs of a flow with values $x_1, \ldots, x_n$. For each X we computed the first order entropy (*Entropy*) and an estimate of the entropy rate (*EntropyRate*) using the corrected conditional entropy (CCE) [8]. The first order entropy is useful for comparing the shape of distributions of random variables, and the entropy rate is useful for comparing the regularity of time series. For computing the CCE we used equiprobable binning of the data as in [8]. To determine the number of bins Q we performed initial tests and found $Q = 5$ maximised the classification accuracy.

We also computed a feature based on the two-sample Kolmogorov-Smirnov (KS) test, which tests the hypothesis that two samples were drawn from the same distribution. A low KS test statistic means that the distributions are similar whereas a high KS test statistic means the distributions are different. The KS test is applicable to a variety of data with different distributions. Since we need a feature that reflects how different a distribution is from the set of distributions characterising normal traffic, we computed the set of KS test statistics between a data instance (covert or normal) and all instances of normal traffic (excluding tests of a normal instance with itself) and use the mean of all KS statistics (*MeanKS*).

5.2 Machine Learning

Previous research showed that for classification of network traffic the better ML techniques provide similar accuracy, but differ greatly regarding training time and classification speed [15]. We used the C4.5 decision tree classifier [16] – more precisely its implementation in the Waikato Environment for Knowledge Analysis (WEKA) [17], because it had performed well previously [15]. Using a decision tree algorithm also has the advantage that a human can interpret the resulting classifier, although with increasing size of the tree this becomes difficult.

C4.5 selects features in order of maximising an entropy-based gain ratio. The most useful features are always used at the top of the tree and irrelevant features are largely ignored. Hence, C4.5 is not adversely affected by irrelevant features like some other techniques and feature pre-selection is not necessary. C4.5 attempts to avoid over-fitting

by removing some structure from a tree after it was built (tree pruning) [16]. In our experiments we used the default WEKA 3.4.4 settings for all parameters of C4.5.

We evaluate the classification accuracy based on the following metrics. A true positive (TP) is a class instance correctly classified. A false positive (FP) is a non-class member misclassified as class member and a false negative (FN) is a class member misclassified as non-class member. *Precision* is the number of class members classified correctly over the total number of instances classified as class members [17]:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \ . \tag{9}$$

Recall (or *TP rate)* is the number of class members classified correctly over the total number of class members [17]:

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \ . \tag{10}$$

We performed 10-fold cross-validation for each dataset. The data set was randomly divided into k subsets, and the classification was repeated k times. Each time, one of the k subsets was used as the test set and the other $k - 1$ subsets formed the training set. Then the average accuracy across all k trials was computed.

The classification accuracy varies depending on the random covert data, the random number series used for encoding R, and the random selection of instances during the cross-validation. Therefore, we performed the cross-validation 10 times and computed the average accuracy.

5.3 Results

Figure 3 shows the precision and recall of the covert traffic class averaged over all datasets for the basic covert channel proposed in [2, 3] and for sub-band encoding depending on the size of the least significant part l. For small l precision and recall drop to 60% or lower. For larger l they increase substantially, but even for $l = 25$ ms precision is still less than 80%, well below the 96% for the basic covert channel. We found the resulting classifiers rely mainly on EntropyRate, but MeanKS is also used occasionally at the bottom of the tree.

A recall of 70–80% for sub-band encoding still appears high. However, one needs to take into account the large FP rate as indicated by the precision of 60–70%. In reality, even when assuming the warden has already selected some traffic fraction of interest, most flows are normal flows. Thus even a FP rate of only a few percent very effectively masks the covert channel. To explore the detection accuracy at low FP rates we computed the Receiver Operating Characteristic (ROC) curves by using cost-sensitive classification and varying the cost for misclassified instances of either class.

Figure 4 shows the low FP-rate part of the ROC curves for the covert traffic class averaged over all datasets for the basic channel proposed in [2,3] and sub-band encoding with a sub-band size of 5 ms. It shows that sub-band encoding is much harder to detect. For a FP rate of 0.5% (indicated by the vertical dashed line) approximately 80% of the basic channels are detected, whereas for sub-band encoding the detection accuracy is greatly reduced to approximately 9%.

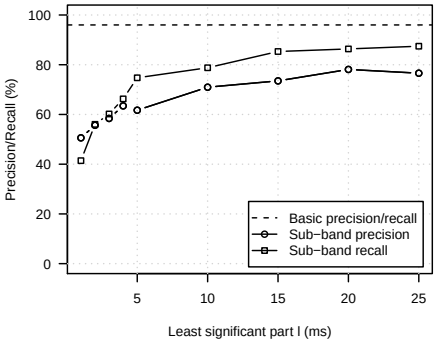

Fig. 3. Precision and recall of covert traffic class for basic encoding and sub-band encoding depending on least significant part l

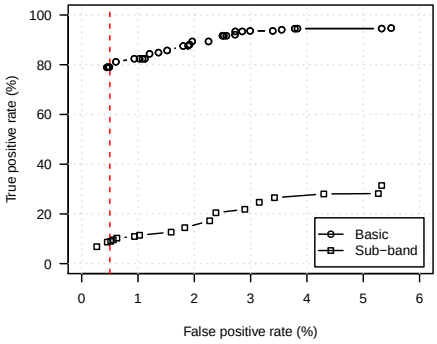

Fig. 4. ROC curves of covert traffic class for the basic covert channel proposed in [2,3] and sub-band encoding with $l = 5$ ms

6 Channel Capacity

We propose a model to compute the channel capacity, describe our experimental setup, and present the measured channel capacities based on different network jitter.

6.1 Channel Model

We assume the channel's output only depends on the input and the errors but not on previous inputs (memoryless channel) and we focus on a binary channel (one bit encoded per IPG). Timing jitter causes bit substitution errors. Under timing jitter we subsume packet timing inaccuracies at the covert sender, timestamping inaccuracies at the covert receiver and network jitter.

In our testbed experiments the resulting error rates were approximately symmetric. Hence we model the channel as Binary Symmetric Channel (BSC) with a capacity [18]:

$$C = 1 - H(p) = 1 + p \cdot \log_2(p) + (1-p) \cdot \log_2(1-p) \ , \tag{11}$$

where $H(.)$ is the binary entropy and p is the probability of timing errors. The capacity C is in bits per IPG (bits per symbol). Given an average rate of IPGs f_S the average transmission rate in bits per second is:

$$R = C \cdot f_S \ . \tag{12}$$

6.2 Testbed and Methodology

We implemented a prototype of sub-band encoding, using the Covert Channels Evaluation Framework (CCHEF) [19], which was carefully designed to maximise Alice's

packet timing accuracy. However, since it is a userspace program it competes with other userspace programs for CPU time. Other programs using a lot of CPU time decrease the timing accuracy. To avoid this we used real-time Linux 2.6.20 and ran Alice and Bob as real-time processes with high priority. We also set the kernel's tick frequency to 10 kHz to reduce the size of time slices.

Our testbed consisted of two computers connected via a Fast Ethernet switch. We used scp to perform file transfers capped at 2 Mbit/s, and SSH to perform remote interactive shell sessions. We generated Q3 with bots as players. Each experiment lasting 20 minutes was repeated three times and we report the average statistics. Network delay and jitter were emulated using Linux Netem [20]. The network delay was emulated using Pareto distributions with a mean of 25 ms and standard deviations (σ) of 0, 0.1, 0.2, 0.3, 0.5, 1 and 2 ms in each direction, since previous research suggested that network jitter is heavy-tailed [21], and Netem only supports Uniform, Gaussian and Pareto jitter distributions. Setting the kernel tick timer frequency to 10 kHz ensured delay emulation was accurate to $\pm 100\,\mu$s.

Figure 5 shows CDFs of the absolute IP Delay Variation (IPDV [22]), both in the testbed with Pareto distributions with different σ and measured across two Internet paths. The 8-hop Internet path's RTT was approximately 32 ms, and the 13-hop path's RTT was approximately 46 ms. We estimated the one-way delay to half the measured RTT. Both Internet path's IPDV CDFs lie between the testbed CDFs for $\sigma = [0.3, 0.5]$.

The IPG models were built as follows. First, we measured the IPG distribution of each application at the source, unaffected by timing jitter. We then added a small amount of noise. Without the added noise the covert channel would not work well for applications with very narrow IPG distribution, such as Q3 client-to-server and scp traffic. The noise represents timing jitter caused by the network, or a high CPU or network interface load of the source host, which the warden would also encounter in reality.

Our models were histograms with small bin sizes of 100 µs, as our traffic sources cannot be modelled well with standard statistical distributions. For Q3 and SSH traffic the location of the sub-bands was chosen such that peaks in the distributions are approximately in the middle of bands.

6.3 Maximum transmission rates

We measured the error rates of the channel based on the network jitter and computed the maximum transmission rates using Equations 11 and 12. Figure 6 shows the maximum rates for sub-band encoding for a sub-band size of $l = 5$ ms. We selected this size because it provides acceptable noise levels for $\sigma \leq 0.5$ ms at a reasonably low detection accuracy (see Section 5). Note that for TCP traffic the channel can only be encoded in one direction due to the timing dependencies between packets in both directions. Multiple TCP flows could be used to achieve full-duplex communication.

Sub-band encoding has significantly higher error rates with Q3 client-to-server traffic or scp compared to Q3 server-to-client traffic or SSH (not shown). However, the transmission rates are still much higher for Q3 client-to-server traffic and scp because of the much higher packet rates. Sub-band encoding has low error rates and reasonably high capacities at low levels of jitter typical of uncongested paths. Transmission rates are up to over hundred bits per second. If the network jitter is high the capacity is

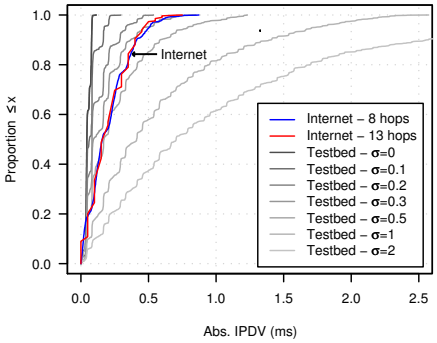

Fig. 5. Absolute IPDV distributions for testbed and two Internet paths

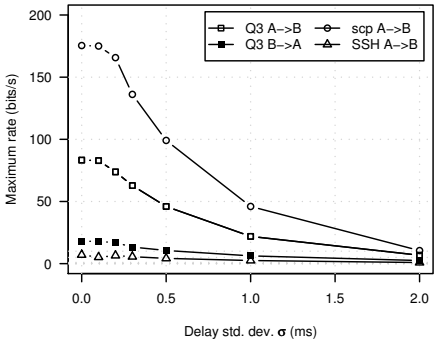

Fig. 6. Maximum transmission rates for sub-band encoding

severely reduced. However, larger sub-band sizes could be used to increase robustness at the cost of reducing the stealth.

Q3 and scp have narrow IPG distributions, and in our experiments even interactive SSH was limited because we played back a short recorded session repeatedly. Applications with wider IPG distributions can provide more robust channels. However, bit rates would still be relatively low since such applications also have lower packet rates.

7 Conclusions and Future Work

We showed that Gianvecchio's and Sellke's IPG timing channel [2, 3] is easy to detect with ~80% accuracy if the IPGs of normal traffic are not iid. Inspired by image steganography we developed an improved encoding scheme for IPG timing channels that is much harder to detect. The detection accuracy is reduced down to ~9%. The random numbers needed for encoding are generated from the packet content, which makes the new channel usable with all UDP-based traffic. Our new channel has reduced robustness, but it still provides a reasonably high channel capacity.

We performed experiments with a proof-of-concept implementation in a testbed with different emulated network jitter. The new channel has low error rates and reasonably high capacities for lower network jitter, comparable to measured jitter on short/medium uncongested Internet paths. Transmission rates are up to over one hundred bits per second, depending on the overt traffic. However, the capacity is severely reduced if the network jitter is high.

In future work our channel model could be extended to include the effects of packet loss. Our analysis could be extended to a wider range of applications generating the overt traffic, and to a larger range of network conditions including packet loss. The sender's timing accuracy could be potentially improved with a future kernel-based implementation. Another avenue for future studies are passive channels, where the covert sender encodes the channel into existing traffic of unwitting hosts.

References

1. S. Zander, G. Armitage, P. Branch. A Survey of Covert Channels and Countermeasures in Computer Network Protocols. *IEEE Communications Surveys and Tutorials*, 9(3):44–57, October 2007.
2. S. Gianvecchio, H. Wang, D. Wijesekera, S. Jajodia. Model-Based Covert Timing Channels: Automated Modeling and Evasion. In *Recent Advances in Intrusion Detection (RAID)*, September 2008.
3. S. H. Sellke, C.-C. Wang, S. Bagchi, N. B. Shroff. Covert TCP/IP Timing Channels: Theory to Implementation. In *Conference on Computer Communications (INFOCOM)*, April 2009.
4. V. Paxson. End-to-end Internet Packet Dynamics. *IEEE/ACM Transactions on Networking*, 7(3):277–292, 1999.
5. M. A. Padlipsky, D. W. Snow, P. A. Karger. Limitations of End-to-End Encryption in Secure Computer Networks. Technical Report ESD-TR-78-158, Mitre Corporation, August 1978.
6. V. Berk, A. Giani, G. Cybenko. Detection of Covert Channel Encoding in Network Packet Delays. Technical Report TR2005-536, Dartmouth College, November 2005.
7. G. Shah, A. Molina, M. Blaze. Keyboards and Covert Channels. In *USENIX Security*, August 2006.
8. S. Gianvecchio, H. Wang. Detecting Covert Timing Channels: An Entropy-Based Approach. In *ACM Conference on Computer and Communication Security (CCS)*, November 2007.
9. X. Luo, E. W. W. Chan, R. K. C. Chang. TCP Covert Timing Channels: Design and Detection. In *IEEE/IFIP Conference on Dependable Systems and Networks (DSN)*, June 2008.
10. Y. Liu, D. Ghosal, F. Armknecht, A.-R. Sadeghi, S. Schulz, S. Katzenbeisser. Hide and Seek in Time – Robust Covert Timing Channels. In *European Symposium on Research in Computer Security*, September 2009.
11. Quake 3. `http://www.idsoftware.com`.
12. P. Branch, A. Heyde, G. Armitage. Rapid Identification of Skype Traffic. In *ACM Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, June 2009.
13. M2C Measurement Data Repository, December 2003. `http://traces.simpleweb.org/`.
14. C. Henke, C. Schmoll, T. Zseby. Empirical Evaluation of Hash Functions for PacketID Generation in Sampled Multipoint Measurements. In *Passive and Active Measurement (PAM) Workshop*, pages 197–206, 2009.
15. N. Williams, S. Zander, G. Armitage. A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification. *SIGCOMM Computer Communication Review*, 36(5), October 2006.
16. R. Kohavi, J. R. Quinlan. *Decision-tree Discovery*, chapter 16.1.3, pages 267–276. Oxford University Press, 2002.
17. I. H. Witten, Eibe Frank. *"Data Mining: Practical Machine Learning Tools and Techniques – 2nd Edition*. Morgan Kaufmann, San Francisco, 2005.
18. T. M. Cover, J. A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley & Sons, 1991.
19. S. Zander. CCHEF - Covert Channels Evaluation Framework, 2007. `http://caia.swin.edu.au/cv/szander/cc/cchef/`.
20. Linux Foundation. Netem, 2008. `http://www.linuxfoundation.org/en/Net:Netem`.
21. L. Rizo, D. Torres, J. Dehesa, D. Muñoz. Cauchy Distribution for Jitter in IP Networks. In *International Conference on Electronics, Communications and Computers*, pages 35–40, 2008.
22. C. Demichelis and P. Chimento. IP Packet Delay Variation Metric for IP Performance Metrics (IPPM). RFC 3393, IETF, November 2002. `http://www.ietf.org/rfc/rfc3393.txt`.